VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA ZABEZPEČENÍ WEBOVÝCH SERVERŮ A DOMÉN
ANALYSIS OF WEBSERVER AND DOMAIN SECURITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Dvonč**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Ilgner**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Tomáš Dvonč ***ID:*** 173974
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Analýza zabezpečení webových serverů a domén

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi zabezpečení doménových jmen v systému DNS a metodami zajištění autentizace a důvěrnosti přístupu k webovým a emailovým službám. Zaměřte se zejména na technologie, které je možné nasadit na české národní doméně. Prozkoumejte nejpoužívanější webové programové vybavení pro správu webového obsahu a sestavte jednoduchou znalostní bázi známých bezpečnostních zranitelností těchto aplikací.

V praktické části navrhněte a v prostředí OS Linux a programovacího jazyku Go sestavte aplikaci, která bude automatizovaně vyhodnocovat, které zkoumané bezpečnostní mechanismy DNS systému jsou implementovány na jednotlivých doménách ze vstupní databáze domén. Pro tyto domény dále prozkoumejte podporu bezpečnostních mechanizmů zajišťující bezpečný přístup k webovým a emailovým serverům. Detekujte použité programové vybavení webových serverů provozovaných na těchto doménách a na základě vytvořené znalostní báze určete potenciálně zranitelné weby.
Aplikace bude vhodně optimalizována, aby detekční úlohy probíhaly efektivně. Na praktickou část bude navazovat část komentující získané výsledky.

DOPORUČENÁ LITERATURA:

[1] KABELOVÁ A., DOSTÁLEK L. Velký průvodce protokoly TCP/IP a systémem DNS. 5. aktualiz. vyd. Brno: Computer Press, 2008, 488 s. ISBN 978-80-251-2236-5.

[2] CHAPPELL L., TITTEL E. Guide to TCP/IP. 3. ed. Boston, Mass: Thompson/Course Technology, 2007, 742 s. ISBN 1418837555.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: Ing. Petr Ilgner
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cieľom bakalárskej práce bola analýza zabezpečenia webových a e-mailových serverov. Práca sa skladá z troch častí. V prvej časti budú popísané komunikačné protokoly a vysvetlí sa zásadný rozdiel medzi nimi. Druhá časť popisuje históriu vývoja kryptografických protokolov, ktoré šifrujú komunikáciu, umožňujú predchádzať odpočúvaniu, manipulácii a falšovaniu dát. Ďalej je uvedená architektúra zabezpečenia, spôsob, akým sa realizuje komunikácia medzi klientom a serverom, a objasní sa pojem certifikačná autorita. Popíšu sa jednotlivé bezpečnostné mechanizmy, ktoré zabezpečujú spoľahlivú komunikáciu prostredníctvom internetu. Tretia časť je zameraná na vytvorenie programu, ktorý bude automatizovane testovať bezpečnostné prvky priloženej sady domén. Aplikácia bola vytvorená v programovacom jazyku *Golang* za pomoci nástroja *Docker*. Výstupom je štatistika analýzy zabezpečenia a detekcia potenciálnych chýb v zabezpečení. Na základe získaných údajov bude užívateľovi priblížený aktuálny stav zabezpečenia webových a e-mailových serverov.

KĽÚČOVÉ SLOVÁ

HTTP, HTTPS, SSL, TLS protokol, Linux, Golang, Docker

ABSTRACT

The aim of the bachelor thesis was to analyze the security of web and e-mail servers. The work consists of three parts. The first part describes communication protocols and explains the basic difference between them. The second part describes the history of development of cryptographic protocols which are used to encrypt communication, prevent data capture and manipulation. Furthermore, the security architecture is presented, the way the client-server communication is implemented and the concept of the certification authority is clarified. In addition, the various security mechanisms that provide reliable communication over the Internet will be described. In the third part the main focus is to create a program that would automatically test the security features of the enclosed domain set. The Application was created in the Golang programming language using Docker. The result is security analysis statistics and security errors detection. Based on the data obtained, the user will be provided with current the security status of web and email servers.

KEYWORDS

HTTP, HTTPS, SSL, TLS protocol, Linux, Golang, Docker

DVONČ, Tomáš. *Analýza zabezpečení webových serverů a domén*. Brno, 2018, 57 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Petr Ilgner

VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Analýza zabezpečení webových serverů a domén“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi Ing. Petrovi Ilgnerovi za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

V dnešnej dobe ľudia využívajú internet na rôzne činnosti, ako napríklad nakupovanie, e-mailovanie, odosielanie dát a podobne. Mnoho stránok požaduje od užívateľov poskytnutie ich citlivých dát, ako sú kreditné karty alebo osobné informácie pri registrácií. Preto je dôležité, aby tieto odosielané dáta neboli zneužité neoprávnenou osobou a bola zaručená integrita správy. Z toho dôvodu je nevyhnutné, aby prenášané dáta z bodu A do bodu B boli šifrované a bol zabránený prístup k ich modifikácií.

Ako však zistiť, ktorá webová stránka je bezpečná, a ktorá nie? Našťastie existuje niekoľko spôsobov, ktorými sa dá skontrolovať, či je web dostatočne zabezpečený. V bakalárskej práci je bližšie venovaná pozornosť k detekcii potenciálnych chýb na webových a emailových serveroch.

V prvej kapitole je popísaný podstatný rozdiel medzi stránkami, ktoré používajú komunikačné protokoly HTTP a HTTPS. Základom každej komunikácie prostredníctvom internetu by malo byť použitie šifrovanej komunikácie. Avšak neskôr sa dozvieme, že v skutočnosti to nie je vždy tak.

Ďalšia kapitola popisuje detailnejšie, akým spôsobom funguje zabezpečenie komunikácie. Rozoberú sa všetky bezpečnostné mechanizmy, ktoré sa podieľajú na prenose informácií medzi klientom a serverom. Pre ich pochopenie je dôležité si objasniť pojmy SSL a TLS, ktoré zabezpečujú šifrované komunikačné spojenie. Na začiatku sa venuje pozornosť historickému vývoju týchto protokolov a nevyhnutnosti naďalej zlepšovať bezpečnostné mechanizmy. Ďalej sú popísané všetky potenciálne nebezpečné a rizikové prvky zabezpečenia, ktoré môžu ohroziť proces komunikácie. Podrobne sa rozoberie architektúra zabezpečenia, od nadviazania komunikácie až po jej samotné ukončenie, a nakoniec bude vysvetlený proces overenia totožnosti organizácie ako certifikačnej autority.

Posledná časť práce sa zaoberá vytvorením automatizovaného nástroja na testovanie zabezpečenia webových a e-mailových serverov v operačnom systéme Linux. Aplikácia je realizovaná v programovacom jazyku Golang a integrovaná v nástroji Docker. Tento nástroj podstatne zjednoduší spustenie programu na ľubovoľnom zariadení. Za pomoci vytvorenej aplikácie je užívateľ schopný testovať vybranú sadu domén. Navyše nástroj bol optimalizovaný do takej miery, aby umožnil analyzovať väčšie množstvo dát v relatívne krátkom čase.

Z výsledkov bude možné vyvodiť záver, ktorý užívateľovi poskytne informácie o aktuálnom stave zabezpečenia webových a e-mailových serverov. Následne sa poukáže na najčastejšie bezpečnostné chyby, ktoré v súčastnosti naďalej pretrvávajú medzi niektorými servermi.

1 Komunikačné protokoly

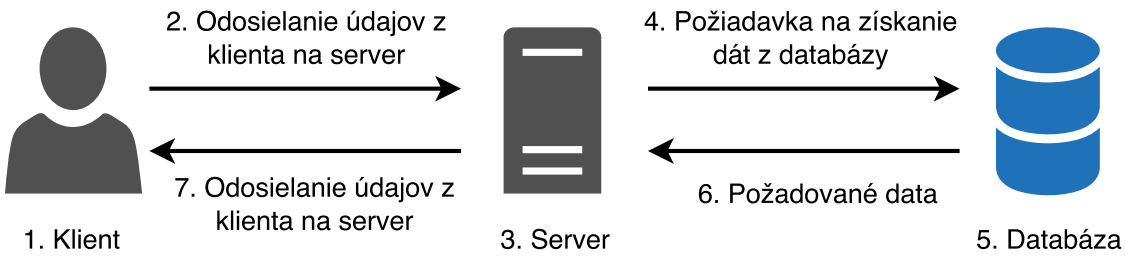
V telekomunikačných sieťach je **komunikačný protokol** systém pravidiel, ktoré umožňujú dvom alebo viacerým zariadeniam prenášať určitú informáciu. Každý protokol popisuje určitú sémantiku, syntax a synchronizáciu komunikácie. Komunikačný protokol môže byť implementovaný do *hardwaru* alebo *softwaru*. Medzi najznámejšie protokoly patria: FTP (*File Transfer Protocol*), TCP/IP, UDP (*User Datagram Protocol*), HTTP (*Hypertext Transfer Protocol*) a ďalšie iné [1].

1.1 HTTP protokol

HTTP je protokol **aplikačnej** vrstvy, slúžiaci na prenos informácií prostredníctvom internetu. Tento protokol používa na komunikáciu port 80. Každý prenášaný dokument môže obsahovať rôzne podklady, ako sú napr. text, obrázky, videa a ďalšie.

Dôležitým aspektom HTTP protokolu je, že neexistuje žiadne prepojenie medzi dvoma rôznymi požiadavkami, ktoré sa uskutočňujú na rovnakom spojení. Server po odpovedi klientovi ukončí pripojenie.

HTTP protokol pracuje štandardne s protokolom TCP. Je to protokol transportnej vrstvy, ktorého základnou podstatou je nadviazanie spojenia, prenos dát a ukončenie spojenia. Základom každej webovej komunikácie je **žiadosť** (*request message*), ktorá je odosielaná prostredníctvom URL (*Uniform Resource Locator*), a **odpoveď** (*response message*), ktorá sa vráti klientovi zo servera, ako odpoveď na jeho požiadavku (viac obr. 1.1).



Obr. 1.1: Prenos informácií klient-server

Hlavná nevýhoda protokolu HTTP spočíva v tom, že odosielaná správa je obyčajný text, nie je šifrovaná, a teda útočník ju môže veľmi jednoducho napadnúť alebo podvrhnúť.

Najčastejší problém predstavuje útok MITM (*Man in the middle*). Princíp útoku je nasledovný: tretia osoba – útočník, sa postaví medzi klienta odosielajúceho správu a server, na ktorý je správa odosielaná. Túto správu dokáže ľahko prečítať, pretože

sa posiela nezašifrovaná, a taktiež ju môže modifikovať. Tento problém bol jedným z dôvodov pre vytvorenie protokolu HTTPS (*Hypertext Transfer Protocol Secure*), ktorý umožňuje šifrovať prenášané dáta cez internet.

Každá komunikácia musí obsahovať minimálne dve zariadenia, medzi ktorými dochádza k prenosu informácií.

1. **Klient**
    - Je to akýkoľvek nástroj, ktorý iniciuje žiadosť a vykonáva úlohu, ktorú vyžiadal používateľ. Tento proces riadi predovšetkým webový prehliadač. V prípade, že bola odoslaná požiadavka na server, webový prehliadač načíta HTML dokument zo stránky.
    - Tento dokument obsahuje rôzne informácie o rozložení (CSS) a spracovaní celého dokumentu. Webový prehliadač potom tieto zdroje prezentuje používateľovi ako webovú stránku.
2. **Webový server**
    - Na druhej strane komunikačného kanálu je server, ktorý príjme požiadavku od klienta a spracuje ju.
    - Odpovedá mu na základe svojich vopred určených pravidiel.

Medzi klientom a serverom môže ešte existovať tzv. **proxy server**, ktorý umožňuje klientom nepriame pripojenie k inému serveru. Funguje ako sprostredkovateľ medzi zariadením koncového bodu (napr. počítač) a iným serverom, od ktorého používateľ alebo klient požaduje službu [2].

1.2 HTTPS protokol

Protokol HTTPS je komunikačný protokol, ktorý používa TCP port 443, čo robí z HTTP a HTTPS protokolu dve rozdielne komunikácie. HTTPS protokol používa systém bezpečnostných certifikátov SSL/TLS, ktoré zabezpečujú šifrovanú komunikáciu prostredníctvom internetu.

Ďalej poskytuje používateľovi **dôvernosť**, **autentifikáciu** servera a **integritu** správ. Toto šifrovanie je mimoriadne dôležité pre webové stránky napríklad elektronického obchodu, kde užívateľ zveruje svoje osobné a platobné údaje, aby nedošlo k úniku dát a krádeži identity. Pre každého užívateľa je dôležité, aby tieto informácie zostali v bezpečí.

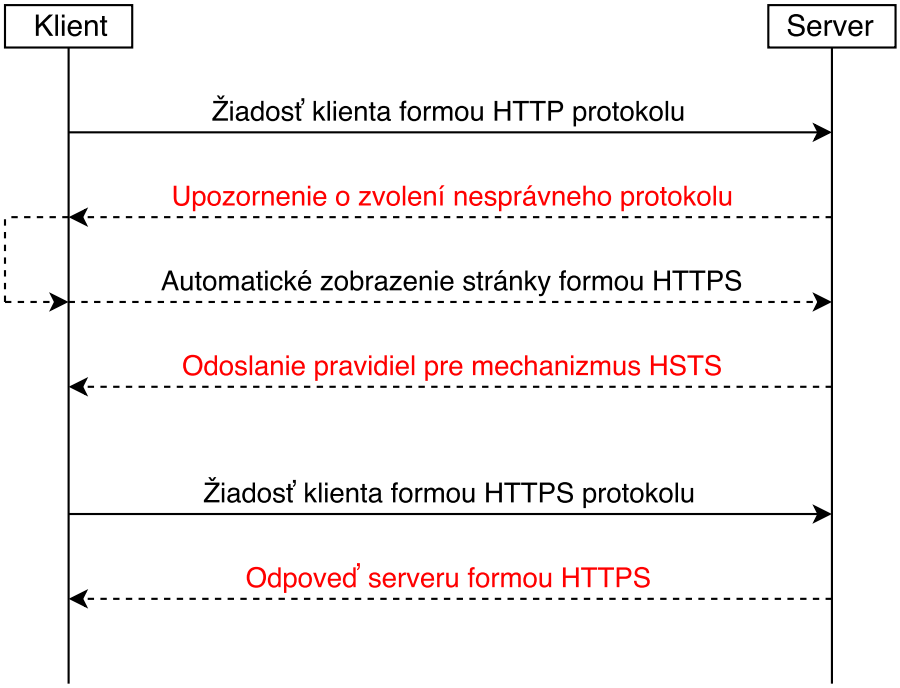
Hneď ako prenášaná informácia dorazí na miesto určenia, server dokáže túto správu spätne dešifrovať do pôvodného formátu [3].

1.3 Bezpečnostný mechanizmus HSTS

HTTP *Strict Transport Security* (HSTS) je **bezpečnostný** mechanizmus, ktorý chráni sieťovú komunikáciu mezi klientom a webovým serverom pred *downgrade* útokmi (zámerné zníženie úrovne šifrovania útočníkom) a zjednodušuje ochranu proti únosu spojenia (tzv. *cookie hijacking*)

Mechanizmus umožňuje, aby webový server vynútil v prehliadači komunikáciu len pomocou šifrovaného HTTPS pripojenia a vylúčil tým prenos dát nezabezpečeným HTTP protokolom (viac obr. 1.2). Webový server aktivuje HSTS pomocou HTTP hlavičky *Strict Transport Security*, ktorá svojou hodnotou definuje časový úsek, počas ktorého môže prehliadač bezpečne pristupovať k serveru.

Ak sa vyskytne chyba pri overení certifikátu (tj. bude sa považovať za nedôveryhodný), prístup sa zablokuje [4].



Obr. 1.2: Štruktúra mechanizmu HSTS

2 Zabezpečenie komunikácie

Táto kapitola popisuje problematiku zabezpečenia webových serverov. Rozoberú sa protokoly, ktoré umožňujú zabezpečiť sieťovú komunikáciu. Na začiatku je popísaný historický vývoj týchto protokolov a poukáže sa na ich nedostatky. Veľmi dôležité je charakterizovanie všetkých prvkov komunikácie, ktoré zohrávajú dôležitú rolu pri zabezpečení. Nakoniec je vysvetlená architektúra komunikácie, od nadviazania až po ukončenie spojenia.

2.1 História vývoja SSL/TLS protokolov

SSL prvýkrát predstavila spoločnosť *Netscape* v rokoch 1993–1994. Rast počtu ľudí používajúcich internet rýchlo stúpal a rovnako aj potreba bezpečnosti komunikácie. Dnes SSL/TLS používa takmer každá online služba.

Prvá verzia **SSL 1.0** nebola nikdy uvedená na trh, pretože mala obrovské bezpečnostné medze. Prvé oficiálne vydanie **SSL 2.0** prišlo o rok neskôr, v roku 1995. Posledná verzia SSL protokolu vyšla v novembri 1996. S odstupom času sa už ani táto verzia nedala považovať za dostatočne zabezpečenú. V roku 2011 bol protokol SSL 2.0 odobraný z trhu, pretože mal tieto nedostatky:

1. Komunikácia používala MD5 (*Message-Digest* 5) hashovací algoritmus, ktorý sa už dal v tej dobe prelomiť *brute force* útokom. Táto forma kryptografického algoritmu sa viac nepovažovala za bezpečnú.
2. Pri nadviazaní komunikácie útočník získal odosielanú správu a bol schopný prinútiť klienta zameniť hashovací algoritmus za zraniteľnejší, aký si v skutočnosti vybral.
3. Integrita a šifrovanie správ používali rovnaký kľúč. V prípade, ak klient a server používali slabý šifrovací algoritmus, dochádzalo k prelomeniu zabezpečenia.
4. Pri komunikácii klienta so serverom bol útočník schopný ukončiť komunikáciu spôsobom, že do nej vložil správu *TCP FIN* na ukončenie relácie, a klient nevedel určiť, či to bol legitímny koniec komunikácie.

V roku 1999 bola uvedená na trh prvá verzia protokolu **TLS 1.0**. Neskôr bola vyvinutá novšia verzia **SSL 3.0**, ale v roku 2015 zastarala – podľa oficiálneho vyhlásenia *IETF* bola akákoľvek verzia TLS považovaná za bezpečnejšiu ako SSL 3.0 [6], a to z nasledujúcich dôvodov:

1. Výmena kľúčov medzi klientom a serverom sa naďalej považovala za nezabezpečenú.
2. Všetky certifikáty boli postavené na princípe hashovacích algoritmov SHA-1 a MD5, ktoré sú považované za slabé.

3. SSL 3.0 malo obmedzené možnosti a nebolo schopné využiť mnohé funkcie, ktoré boli pridané do novších verzií TLS.

TLS 1.1 vyšla v roku 2006 a oproti staršej verzií bola pridaná ochrana proti CBC (*cipher-block chaining*) útokom. Implicitný inicializačný vektor bol nahradený explicitným.

TLS 1.2 bola uvedená na trh v roku 2008 ako pokračovanie protokolu TLS 1.1 s novými zmenami. MD5 a SHA-1 hashovacie algoritmy boli nahradené za novšiu kryptografickú hashovaciu funkciu SHA-256.

TLS 1.3 je v štádiu vývoja a podrobnejšie detaily sú zatiaľ nejasné. Niektoré z hlavných zmien oproti verzii TLS 1.2 budú:

1. Ukončenie podpory MD5 a SHA-224 kryptografických hashovacích funkcií.
2. Zavedenie vyžadovanie digitálneho podpisu aj v prípade použitia predchádzajúcej konfigurácie.
3. Podpora znižovania počtu nezabezpečených alebo zastaraných funkcií vrátane kompresie a ďalších iných [5].

2.2 Kryptografické protokoly

Šifrovanie je proces, v ktorom sa čitateľná správa (text) konvertuje na **šifrovaný** formát, ktorý nie je človekom čitateľný. Hlavným účelom šifrovania je zabezpečiť, aby iba autorizovaná strana mohla dešifrovať a čítať pôvodnú správu. Keď sa nezašifrované dáta vymieňajú medzi dvoma účastníkmi, s použitím akéhokoľvek média, tretia strana môže zachytiť a prečítať vymieňanú komunikáciu.

Ak výmena obsahuje citlivé informácie, a tretia strana môže takúto komunikáciu zachytiť, mohla by tiež manipulovať s údajmi, prípadne zmeniť informácie, ktoré sa vymieňajú, čím ohrozí integritu správy.

Jeden zo spôsobov, ako zmierniť potenciálny útok a zabezpečiť komunikáciu, je použitie kryptografického protokolu SSL alebo TLS, ktoré budú detailnejšie vysvetlené v nasledujúcej časti.

2.2.1 SSL protokol

SSL (*Secure Sockets Layer*) je kryptografický protokol, ktorý na prenos informácií využíva protokol transportnej vrstvy TCP.

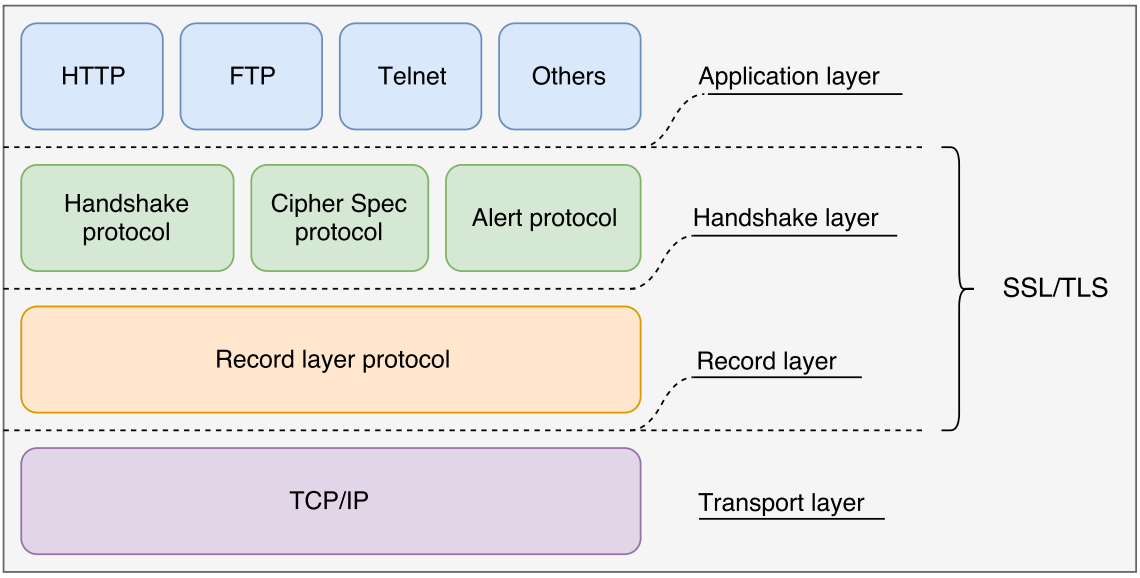
Základnou podstatou protokolu SSL je poskytnúť súkromie a spoľahlivosť medzi dvomi komunikujúcimi aplikáciami.

SSL používa kombináciu verejného kľúča a symetrického šifrovania kľúčov na zabezpečenie spojenia medzi dvoma zariadeniami. Zvyčajne sa jedná o webový alebo

poštový server, komunikujúci prostredníctvom internetu alebo inej siete, napríklad TCP/IP.

SSL poskytuje mechanizmus na šifrovanie a overovanie údajov pri prenose informácií medzi klientom a serverom. SSL pracuje medzi transportnou a sieťovou vrstvou, ktoré sú zodpovedné za prenos a smerovanie dát pod protokolmi aplikačnej vrstvy, ako je HTTP a SMTP (*Simple Mail Transport Protocol*).

Okrem podpory prenosu informácií medzi webovým serverom a klientom je tento protokol implementovaný aj pre aplikácie vrátane e-mailu, prenosu súborov, správ a hlasu (VoIP). Na obrázku 2.1 je znázornená štruktúra protokolu SSL/TLS.



Obr. 2.1: Štruktúra SSL/TLS protokolu

Protokol SSL obsahuje dva podprotokoly.

1. ***Record* protokol:** definuje spôsob, akým si komunikujúce zariadenia vymieňajú dáta pomocou protokolu SSL. Ďalej špecifikuje, ako majú byť odosielané údaje pripravené na prenos, ako majú byť overené a dešifrované pri prijatí.
2. ***Handshake* protokol:** definuje, ako klient a server vytvoria spojenie SSL, vrátane dohody o tom, ktoré kryptografické systémy budú použité.

Z dôvodu početných nedostatkov a zraniteľnosti protokolu a implementácie bol protokol SSL od roku 2015 považovaný organizáciou *IETF* (*Internet Engineering Task Force*) za zastaraný, a preto bol nahradený protokolom TLS, ktorý je spätne kompatibilný s protokolom SSL 3.0 [6].

2.2.2 TLS protokol

S cieľom poskytnúť nový internetový štandard SSL, vydal *IETF* v januári 1999 protokol TLS (*Transport Layer Security*) [7], zabezpečujúci integritu údajov a poskytovanie súkromia medzi dvomi komunikujúcimi stranami. V súčasnosti je to najrozšírenejší bezpečnostný protokol, používa sa pre webové prehliadače a iné aplikácie, ktoré vyžadujú bezpečnú výmenu dát cez sieť, ako sú prenosy súborov, VPN pripojenia a VoIP.

Protokol TLS je navrhnutý tak, aby aplikáciám typu klient-server umožňoval komunikovať bezpečným spôsobom: autentifikáciou, predchádzaním odposluchu a odolnosťou proti modifikácii správ. V praxi zabezpečená webová aplikácia využíva všetky tri služby.

1. **Šifrovanie**: mechanizmus na utajenie toho, čo sa posiela z jedného hostiteľa do druhého
2. **Autentifikácia**: mechanizmus na overenia pravosti poskytnutého identifikačného materiálu.
3. **Integrita**: mechanizmus na detekciu neoprávnenej správy a falšovania správ.

Za účelom vytvorenia kryptograficky zabezpečeného dátového kanálu sa musia účastníci dohodnúť, ktoré šifry a kľúče budú používať na šifrovanie dát. Protokol TLS špecifikuje dobre definovanú sekvenciu nadviazania komunikácie na vykonanie tejto výmeny (viac v časti 2.4.3). Súčasťou nadviazania komunikácie je poskytnutie spôsobu overenia svojej totožnosti. Pri používaní v prehliadači tento mechanizmus autentifikácie umožňuje klientovi overiť, či je server (napr. banka) ten, kto tvrdí, že je, a nie je niekto iný. Tým sa overí totožnosť organizácie. Okrem toho môže server tak isto voliteľne overiť totožnosť klienta – napr. firemný proxy server môže overiť všetkých zamestnancov, z ktorých každý môže mať svoj vlastný jedinečný certifikát podpísaný spoločnosťou.

Nakoniec pri šifrovaní a autentifikácii protokol TLS poskytuje aj svoj vlastný mechanizmus šifrovania správ a každú správu podpíše autentifikačným kódom správy MAC (*Message Authentication Code*). MAC algoritmus je jednosmerná kryptografická hashovacia funkcia, ktorej kľúče sú predurčené oboma partnermi. Pri každom odoslaní záznamu TLS sa pre danú správu vygeneruje, pripojí hodnota MAC a prijímač potom dokáže vypočítať a overiť odoslanú hodnotu MAC na zabezpečenie integrity a pravosti správ.

V kombinácií tieto tri mechanizmy slúžia ako základ pre bezpečnú komunikáciu na webe. Všetky moderné webové prehliadače poskytujú podporu pre rôzne šifry. Dokážu autentifikovať klientov, servery a transparentne vykonávať kontrolu integrity správ pre každý záznam [8].

2.3 Kryptografické algoritmy

Úroveň zabezpečenia každého servera s podporou protokolu HTTPS závisí od zvoleného typu šifrovacej sady, ktorá špecifikuje bezpečnostné mechanizmy. Tieto mechanizmy sú väčšinou kryptografické algoritmy a funkcie, ktoré sa starajú o dôveryhodný prenos dát.

V tejto sekcii sa detailnejšie popíšu jednotlivé časti šifrovacej sady, hashovacie funkcie a poukáže sa na ich zraniteľné verzie.

2.3.1 Hashovacie funkcie

Funkcia hash je ľubovoľná funkcia, ktorú je možné použiť na mapovanie údajov ľubovoľnej veľkosti na údaje s pevnou veľkosťou.

Hashovacie funkcie sú vo všeobecnosti používané v počítačových systémoch na transformáciu veľkého množstva dát do čísel veľkého rozsahu. Kryptografická hashovacia funkcia sa používa na ochranu proti úmyselnému poškodeniu dát v kryptografických aplikáciách. Tradične sú tieto funkcie používané ako štrukturovaný mechanizmus pri ukladaní informácií do dátových úložísk, pre rýchle vyhľadávanie.

Sú tiež užitočné v kryptografii, kde sa dá ľahko overiť, či niektoré vstupné dáta mapujú danú hodnotu hash. V prípade, ak nie sú známe vstupné dáta, je zámerne ťažké ju rekonštruovať znalosťou uloženej hodnoty hash. Toto sa využíva na zabezpečenie integrity prenášaných dát a je stavebným blokom pre HMAC (*hash-based message authentication code*), ktoré poskytujú autentifikáciu správ.

Vzhľadom na to, že hash funkcie sú odolné voči kolízii, šanca prístupu k systému s nesprávnym heslom je extrémne nízka.

MD (*Message Digest*) tvorí skupinu hashovacích algoritmov navrhnutých profesorom *Ronaldom R. Rivestem* z *Massachusettského inštitútu technológií*. Keď bol analytikmi vyhlásený hashovací algoritmus MD4 za nedostatočne zabezpečený, bol v roku 1991 vytvorený algoritmus MD5 ako jeho bezpečná náhrada. **MD5** algoritmus je navrhnutý tak, aby bol pomerne rýchly na 32-bitových strojoch. Okrem toho tento algoritmus nevyžaduje veľké substitučné tabuľky a môže byť kódovaný celkom kompaktne.

Algoritmus MD5 je široko používaná hash funkcia produkujúca 128-bitovú hodnotu hashu. Napriek tomu, že MD5 bol pôvodne navrhnutý na použitie ako šifrovacia funkcia, bolo zistené, že trpí rozsiahlou zraniteľnosťou [9].

Bezpečnosť MD5 bola vážne ohrozená, pričom jej slabé miesta boli odhalené. Najznámejší je *flame malware* z roku 2012. Inštitút softvérového inžinierstva CMU považuje MD5 za v podstate kryptograficky „rozbitý“ a nevhodné na ďalšie použitie. Napriek tejto známej zraniteľnosti sa MD5 stále používa [10].

Secure Hashing Algorithms, tiež známy ako **SHA**, je skupina kryptografických funkcií navrhnutých tak, aby uchovávali údaje zabezpečené. Funguje spôsobom, že transformuje dáta pomocou algoritmu, ktorý pozostáva z bitových, modulárnych operácií a kompresných funkcií. Funkcia hash potom vytvorí pevnú veľkosť reťazca, ktorý sa významne líši od originálu.

Tieto algoritmy sú riešené tak, aby boli jednosmerné, čo znamená, že akonáhle sa transformujú na ich príslušné hodnoty hash, je prakticky nemožné ich premeniť späť na pôvodné dáta.

SHA-1 bol vyvinutý v roku 1993 americkou vládnou agentúrou pre normalizáciu, Národným inštitútom pre štandardy a technológie (*NIST*). Je široko používaný v bezpečnostných aplikáciách a protokoloch (napr. SSL, TLS). SHA-1 pracuje tak, že posiela správu ako bitový reťazec s dĺžkou menšou než 2^{64} bitov a produkuje 160-bitovú hodnotu hash známu ako správa *digestu*.

V dôsledku známych zraniteľností SHA-1 bol tento algoritmus modifikovaný a upravený na **SHA-2**, ktorý pozostáva nie z jednej, ale z dvoch hashových funkcií, známych ako SHA-256 a SHA-512, s použitím 32 a 64-bitových slov.

Existujú ďalšie verzie týchto hashových funkcií, ktoré sú známe ako SHA-224, SHA-384, SHA-512/224 a SHA-512/256, ktoré môžu byť použité pre niektorú časť algoritmu.

Útoky na SHA-2 nie sú také účinné ako proti SHA-1. Hrubá sila na odhalenie takejto správy by si vyžadovala časovo nerealizovateľné výpočty, čo zabezpečuje, že SHA-2 je oveľa bezpečnejší voči týmto druhom útokov [11].

2.3.2 Šifrovacia sada

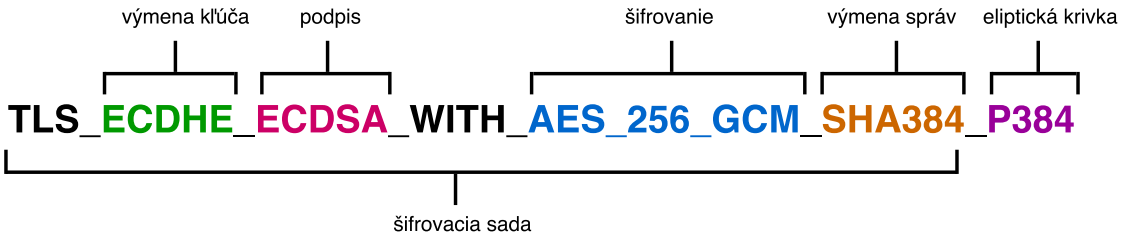
Šifry sú algoritmy zodpovedajúce za vykonávanie šifrovania ako aj dešifrovania. V súčasnosti sú šifry závislé na pokročilých možnostiach spracovania. To však nie vždy bola pravda. Jedna z prvých známych historických šifier patrila *Caesarovi* – cisárovi Ríma, ktorý ho používal na komunikáciu so svojimi generálmi počas vojenských operácií. Okrem toho v druhej svetovej vojne sa využívala na komunikáciu medzi vojakmi – Enigma, ktorá bola prenosným elektromechanickým šifrovacím strojom. Myšlienkou bolo vždy zakódovanie alebo šifrovanie správy takým spôsobom, že ju môže čítať len určená strana.

Existuje veľa rôznych šifier, ktoré sa bežne používajú v spojení s ostatnými šiframi. Je to preto, aby sa v prípade protokolov SSL/TLS nepoužíval len jeden algoritmus, ale skôr súbor algoritmov, ktoré sú zoskupené do celku, čo sa označuje ako **šifrovacia sada**.

Vzhľadom na to, že je veľa počítačov s rôznymi verziami operačných systémov, ktoré používajú niekoľko typov prehliadačov, musí existovať spôsob, ako každému

umožniť použitie optimálnych šifier. To umožňujú práve šifrovacie sady.

Webový server a prehliadač vytvárajú spojenie a môžu následne porovnať zoznam svojich šifrovacích balíkov. Dôležitý je výber vhodnej a zároveň kompatibilnej šifrovacej sady pred nadviazaním komunikácie medzi klientom a serverom. Príklad takejto šifrovacej sady môžeme vidieť na obr. 2.2.



Obr. 2.2: Štruktúra šifrovacej sady

Šifrovacie sady môžu obsahovať algoritmy pre následujúce procesy:

1. **Výmena kľúčov** – umožňuje obom stranám výmenu kryptografických kľúčov. Ak si odosielateľ a prijímateľ chcú vymieňať šifrované správy, musia byť vybavení algoritmom na šifrovanie správ, ktoré sa majú odosielať a následne dešifrovať. Medzi typické algoritmy výmeny kľúčov patrí napr. **RSA**, bežný asymetrický kryptosystém, ktorý využíva prvočísla, použiteľný pre široké spektrum aplikácií. Okrem RSA je dosť používaný ***Diffie-Hellman*** protokol alebo ***Elliptic Curve Diffie-Hellman***, ktorý umožňuje obom stranám vytvoriť spoločný zdieľaný kľúč. Nakoniec treba spomenúť **PSK** (*Pre-shared key*), ktorý je zdieľaný medzi oboma stranami pomocou bezpečného kanála ešte predtým, než sa musí použiť.
2. **Digitálny podpis** – poskytuje zabezpečenie dôvernosti, integrity a overovanie údajov v rámci jedného rozhrania. Algoritmy, ktoré zabezpečujú tento proces sú **RSA**, **DSA** (*Digital Signature Algorithm*) alebo **ECDSA** (*Elliptic Curve Digital Signature Algorithm*). Čo sa týka algoritmu DSA, generovanie kľúčov má dve fázy. V prvej sa vyberú algoritmické parametre, ktoré môžu byť zdieľané medzi rôznymi používateľmi systému, zatiaľ čo v druhej fáze sa vypočítajú verejné a súkromné kľúče pre jedného používateľa.
3. **Šifrovanie** – Medzi najznámejšie šifry, ktoré sa v súčastnosti používajú, patria napr. **RC4**, **AES**, **DES**.
4. **Výmena správ** – používa hashovacie funkcie pre autentifikáciu správy a zabezpečenie integrity údajov. Najpoužívanejšie hashovacie algoritmy sú **SHA** a **MD5** [12].

2.4 Architektúra zabezpečenia (SSL/TLS)

Kryptografia je nevyhnutný nástroj na ochranu informácií v počítačových systémoch. Používa ju denne milión ľudí na celom svete. Kryptografické systémy sú najdôležitejšou súčasťou štandardných protokolov, ako je napríklad **TLS**, čo pomerne ľahko začleňuje silné šifrovanie do širokej škály aplikácií.

Kryptografia je mimoriadne užitočná, avšak je tiež veľmi krehká. Aj najbezpečnejší kryptografický systém môže byť prelomený jedinou zlou špecifikáciou alebo programovou chybou. Ani veľké množstvo testovania nedokáže odhaliť potenciálnu zraniteľnosť zabezpečenia v kryptografickom systéme. Namiesto toho sa spoliehame na matematické modelovanie a dôkazy, že určitý systém spĺňa bezpečnostné vlastnosti, ktoré mu boli pridelené [13].

2.4.1 Výmena kľúčov

Je to akákoľvek metóda kryptografie, ktorou sa vymieňajú kryptografické kľúče medzi dvoma stranami, čo umožňuje následne používať kryptografický algoritmus. Dôležitá vlastnosť, ktorá určuje účinnosť šifry, je veľkosť tajného kľúča. Čím je kľúč väčší, tým zložitejšie je prelomenie kódu. Pre pochopenie uvedieme algoritmus s extrémne malou veľkosťou kľúča: dva bity. V tomto prípade samotný typ algoritmu naozaj nezohráva dôležitú úlohu. S dvomi bitmi existujú len štyri možné kombinácie kľúčov. Útočník, ktorý získal šifrované údaje, jednoducho môže vyskúšať všetky štyri možnosti.

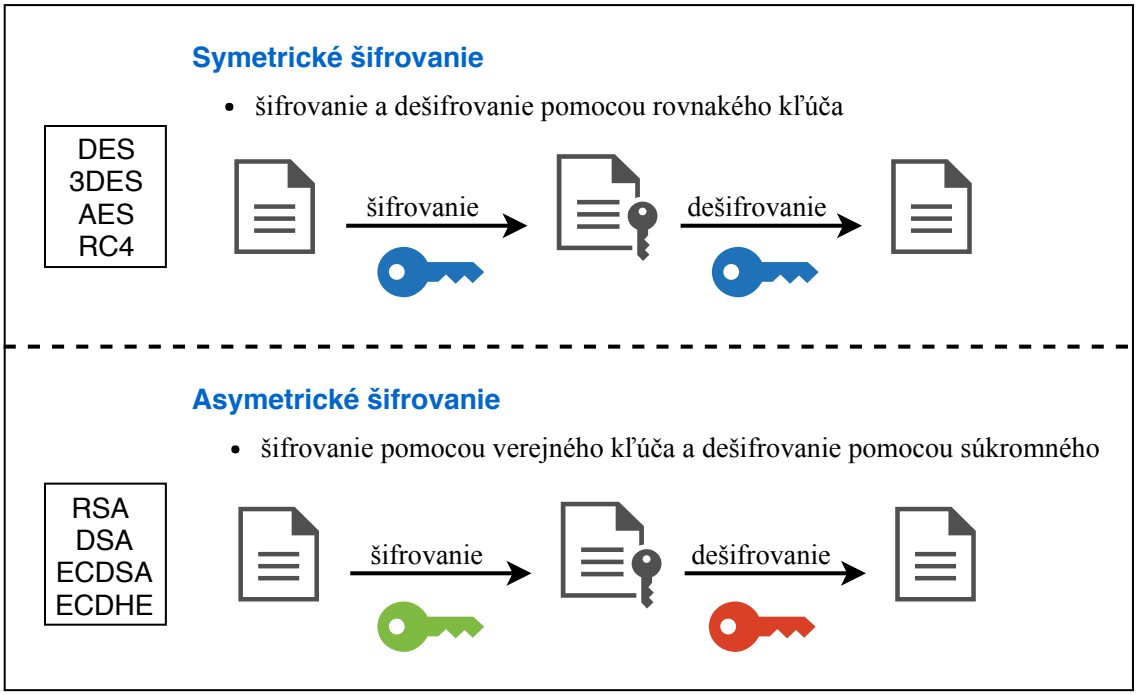
Existujú dve základné techniky šifrovania informácií: **symetrické** šifrovanie (nazývané šifrovanie tajného kľúča) a **asymetrické** šifrovanie (šifrovanie verejným kľúčom).

Symetrické šifrovanie je najstaršou a najznámejšou používanou technikou. V texte správy sa použije **tajný kľúč**, aby sa obsah zmenil určitým spôsobom. Použije sa napríklad číslo, slovo alebo len reťazec náhodných znakov. Môže to byť aj jednoduché posunutie každého písmena o niekoľko miest v abecede. Pokiaľ odosielateľ aj príjemca poznajú tajný kľúč, môžu šifrovať a dešifrovať všetky správy, ktoré tento kľúč využívajú. Kryptografovia tiež charakterizujú symetrické šifrovacie algoritmy podľa toho, ako spracovávajú vstupné dáta. Môžeme ich rozdeliť do dvoch skupín:

- **Prúdové šifry** spracovávajú vstupné dáta (bajty) v čase a môžu prijať akúkoľvek veľkosť vstupu pre šifrovanie.
- **Blokové šifry** na rozdiel od toho pracujú iba na blokoch dát s pevnou veľkosťou, typicky 8 bajtov. V porovnaní s prúdovými šiframi vyžadujú menej výpočtov a sú vo všeobecnosti o niečo menej náchylné na útok.

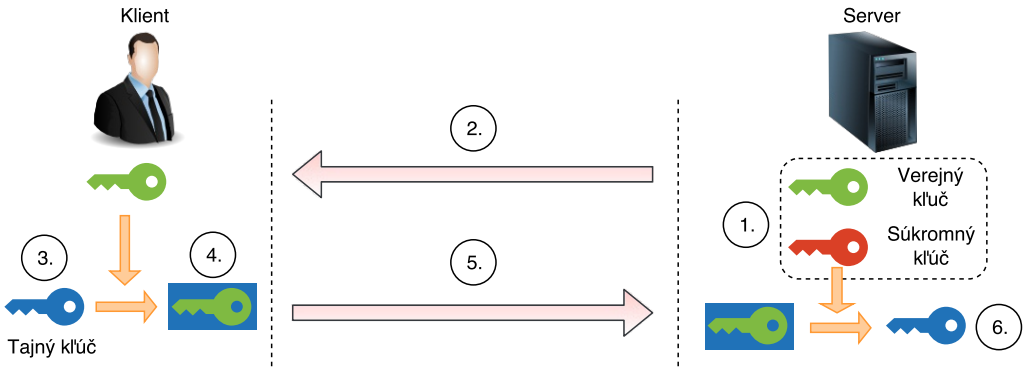
Asymetrické šifrovanie vzniklo ako riešenie problému symetrického šifrovania, ktoré používa jeden kľúč na výmenu informácií cez internet. Každý, kto pozná tento kľúč, môže dešifrovať správu. Asymetrické šifrovanie používa pri výmene informácií dva kľúče, **verejný** kľúč a **súkromný** kľúč. Verejný je k dispozícii každému, kto sa snaží odoslať nejakú správu. Druhý, súkromný kľúč, je uchovaný v tajnosti. Každá správa (text, binárne súbory alebo dokumenty), ktorá je zašifrovaná pomocou verejného kľúča, sa môže dešifrovať iba s použitím rovnakého algoritmu, ale zároveň s použitím zodpovedajúceho súkromného kľúča.

Asymetrické šifrovanie má vo väčšine praktických príkladov jednu vážnu nevýhodu – šifrovanie je veľmi zložité. Toto šifrovanie je založené na tzv. jednocestných funkciách, čo sú operácie, ktoré sa ľahko realizujú len v jednom smere. Zo vstupu sa jednoducho dá spočítať výstup, avšak z výstupu je veľmi zložité nájsť vstup. Najbežnejším príkladom je násobenie. Je celkom jednoduché vynásobiť dve veľmi veľké čísla, ale rozklad súčinu na činitele (tzv. faktorizácia) je veľmi obtiažny. Komplexné matematické operácie môžu na niektoré systémy vyvinúť zaťaženie, čo si vyžaduje väčšiu kapacitu. Našťastie existuje pomerne jednoduchý spôsob, ako získať výhody šifrovania verejného kľúča, a zároveň sa vyhnúť väčšine nákladov na systém. Na obrázku 2.3 je znázornený základný rozdiel medzi symetrickým a asymetrickým šifrovaním.



Obr. 2.3: Porovnanie symetrického a asymetrického šifrovania

Hybridný kryptosystém je optimálny spôsob, ako zlepšiť výkonnosť a efektivitu výmeny informácií. Je to kombinácia symetrického a asymetrického šifrovania. V podstate sa jedna o to, že pre veľmi dlhé správy je väčšina práce v šifrovaní a dešifrovaní vykonaná efektívnejšou schémou symetrických kľúčov, zatiaľ čo neefektívna schéma asymetrického šifrovania sa používa iba na šifrovanie a dešifrovanie krátkej kľúčovej hodnoty. Všetky praktické implementácie kryptografie s asymetrickým šifrovaním používajú dnes hybridný kryptosystém. Na obrázku 2.4 je znázornená architektúra hybridnej komunikácie.



Obr. 2.4: Efektívna ochrana pomocou hybridného kryptosystému

Postupnosť realizácie hybridnej komunikácie je nasledovná:

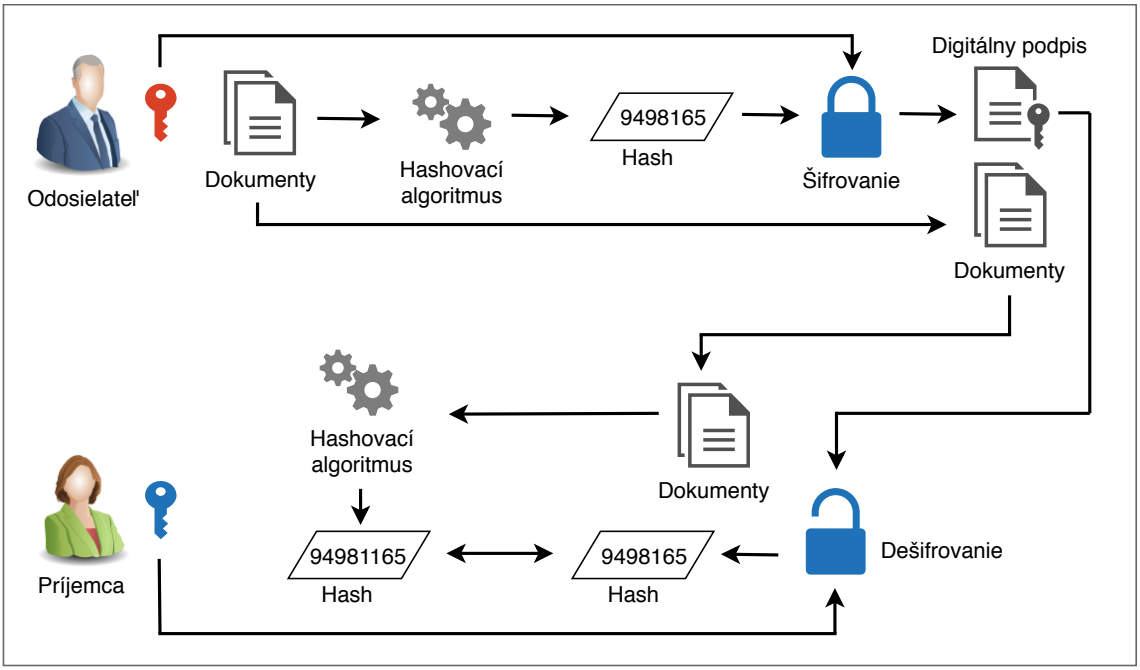
1) Server na začiatku komunikácie vytvorí verejný a súkromný kľúč.
2) Vytvorený verejný kľúč je zdieľaný s užívateľom. Súkromný kľúč si server ponecháva v utajení.
3) Užívateľ si vygeneruje svoj vlastný **tajný** kľúč.
4) Pomocou verejného kľúča zašifruje svoj tajný.
5) Zašifrovaný tajný kľúč následne odošle na server.
6) Server pomocou súkromného kľúča dešifruje správu a získa tak užívateľov tajný kľúč. Po ukončení procesu obe strany vlastnia unikátny tajný kľúč a komunikácia môže začať [14].

2.4.2 Digitálny podpis

Digitálny podpis je matematická schéma na preukázanie pravosti digitálnych správ alebo dokumentov. V jednoduchosti povedané, je to elektronický odtlačok prsta. Umožňuje podpísať dokument elektronicky a zároveň overiť podpisovateľa. Ide o matematický kód, ktorý autentifikuje dokument od odosielateľa a zabezpečuje, aby dokument zostal nezmenený u príjemcu. V dnešnej dobe je digitálny podpis pomerne často využívaný a poskytuje niekoľko výhod:

1. **Šetrí čas** – jedna z obrovských predností digitálnych podpisov je, že umožňujú spoločnosti jednoduchým kliknutím na tlačidlo ušetriť náklady na čas spojený s podpismi dokumentov a zmlúv. Dokumenty je možné podpísať takmer okamžite, odkiaľkoľvek. Či už je to tablet, telefón alebo počítač, digitálne podpisy dokážu bez problémov zaistiť, že táto inak zdĺhavá úloha bude vybavená v priebehu niekoľkých minút.
2. **Úspora nákladov** – mnohým spoločnostiam poskytuje značné úspory nákladov, bez výdavkov na atrament, papier, tlač, skenovanie, prepravu alebo cestovné výdavky.
3. **Efektívnosť pracovného toku** – s menším zdržaním zaručujú digitálne podpisy vyššiu efektivitu v pracovnom procese. Správa a sledovanie dokumentov sa riadi s menším úsilím a časom. Mnoho funkcií digitálnych podpisov pomáha urýchliť pracovný proces. Napríklad e-mailové upozornenia pomáhajú pripomenúť osobe, aby sa podpisovala, zatiaľ čo sledovanie stavu pomáha zistiť, v ktorej fáze je dokument.
4. **Lepšie skúsenosti zákazníkov** – digitálne podpisy poskytujú pohodlie podpisovania dôležitých dokumentov všade, kde sa nachádza zákazník alebo osoba, ktorá podpisuje. Predajcovia nemusia čakať, kým zákazník príde do banky alebo do kancelárie. Je to ideálne, najmä v odľahlých oblastiach a menších obciach poskytujúcich zdokonalené a personalizované služby.
5. **Zabezpečenie** – pokiaľ ide o podpisy, autenticita a bezpečnosť sú prioritou. Digitálne podpisy znižujú riziko duplicity alebo zmeny samotného dokumentu. Okrem toho zaobstarávajú autentifikáciu overených a legitímnych podpisov. Podpisovateľom sú poskytnuté kódy PIN, heslá a kódy, ktorými dokážu autentifikovať a overiť svoju totožnosť a schváliť svoje podpisy. Časové označenie poskytuje dátum a čas podpisu, čím sa minimalizuje riziko narušenia alebo podvodu. Bezpečnostné funkcie vložené do digitálnych podpisov zabezpečujú, že dokumenty neboli zmenené bez povolenia.
6. **Právna platnosť** – digitálne podpisy zaisťujú pravosť a overenie podpisu. Takto sa môžu overovať v súdnom konaní ako akýkoľvek iný podpísaný papierový dokument. Časové označovanie, schopnosť sledovať a ľahko archivovať dokumenty zlepšujú, zjednodušujú audit a dodržiavanie predpisov.
7. **Prínosy pre životné prostredie** – keďže si podniky uvedomujú svoju úlohu v oblasti údržby, digitálne podpisy sú krokom vpred v ich úsilí znížiť množstvo odpadu a byť šetrnými k životnému prostrediu.
8. **Efektívnosť podnikania** – náklady na integráciu digitálnych podpisov do pracovných procesov sú relatívne malé v porovnaní s ich prínosmi. S kratšou dobou trvania dokončenia zmluvy a skrátením času pracovného toku sú digitálne podpisy ideálne pre malé aj veľké organizácie [15].

Digitálne podpisy používajú digitálne identifikačné číslo založené na certifikátoch vydaných akreditovanou certifikačnou spoločnosťou (CA) alebo poskytovateľom dôveryhodných služieb TSP (Trust Service Provider). Podpis je viazaný na dokument so šifrovaním a všetko sa dá overiť s použitím základnej technológie, známej ako infraštruktúra verejných kľúčov (PKI). Na obrázku 2.5 môžeme vidieť princíp fungovania digitálneho podpisu.

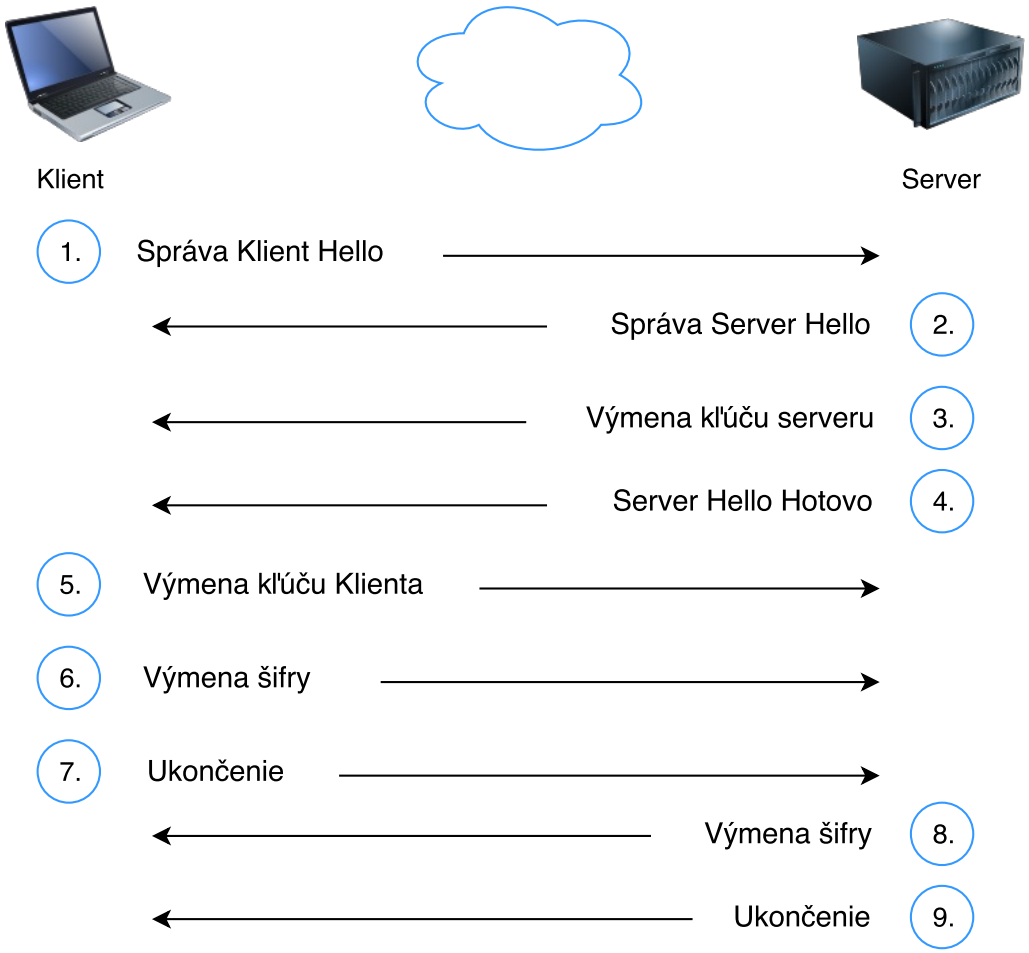


Obr. 2.5: Proces vytvorenia a spracovania digitálneho podpisu

Pred začiatkom komunikácie odosielateľ vygeneruje **verejný** a **súkromný** kľúč. Súkromný kľúč zostane nezverejnený a tajne uschovaný u odosielateľa. Verejný kľúč je odoslaný príjemcovi správy. Potom sa odosielaný dokument zahashuje určeným typom hashovacieho algoritmu. Akonáhle je dokument vo forme hashu, nasleduje šifrovanie dokumentu pomocou asymestrického kryptografického algoritmu. Týmto procesom vznikol **digitálny podpis**. Následne sa digitálne podpísaný dokument a pôvodný dokument odošle príjemcovi. Príjemca po prijatí oboch dokumentov vykoná nasledujúce body. Ako prvé digitálne podpísaný dokument dešifruje pomocou verejného kľúča. V prípade, ak sa podarí dešifrovanie, získa zahashovaný dokument. V druhom kroku pôvodný dokument zahashuje pomocou toho istého hashovacieho algoritmu. V prípade, ak sa oba zahashované dokumenty zhodujú, prijímateľ sa nemusí obávať straty integrity dokumentu.

2.4.3 Nadviazanie komunikácie

Najzákladnejšou funkciou, ktorú môže klient a server vykonať, je vytvorenie kanálu pre šifrovanú komunikáciu. Táto časť bude viac zameraná na postup výmeny správy a podrobnejšie sa preskúma každá správa v rámci výmeny. Obrázok 2.6 zobrazuje proces výmeny správy SSL/TLS, ktorú táto operácia vyžaduje.



Obr. 2.6: Proces vytvorenia šifrovanej komunikácie

1. **Správa Klient Hello** spustí komunikáciu SSL/TLS medzi dvoma stranami. Klient použije túto správu, aby požiadal server o ustanovenie bezpečnostných služieb. Zoznam komponentov správy Klient Hello:
   (a) **Verzia** obsahuje najvyššiu možnú verziu SSL alebo TLS protokolu, ktorú klient podporuje. V prípade, ak klient odošle verziu TLS 1.2 na server, ktorý podporuje iba verziu TLS 1.1, server bude odpovedať klientovi verziou 1.1 . V takejto situácií sa klient môže rozhodnúť pokračovať v komunikácií s verziou TLS 1.1 alebo ju ukončiť.

(b) **Náhodne číslo** je 32 bitové číslo a slúži pre prvotné kryptografické výpočty. Podľa špecifikácie prvé 4 bajty pozostávajú z času a dátumu. Táto špecifikácia naznačuje použitie dátumu a času ako spôsob zabezpečenia, aby klient nikdy nepoužil rovnakú hodnotu dvakrát. Toto opatrenie zabezpečuje ochranu pred falšovateľmi, ktorí využívajú kopírovanie starých SSL správ na vytvorenie falšovanej relácie. Zvyšných 28 bajtov tejto hodnoty je kryptografický zabezpečené náhodné číslo.

(c) **ID relácie** slúži na identifikáciu príslušnej relácie.

(d) **Modely šifier** umožňujú klientovi špecifikovať rôzne kryptografické šifry, vrátane presných algoritmov a veľkosti kľúčov. Server nakoniec rozhoduje o tom, ktorá kryptografická šifra z tohoto zoznamu bude ďalej použitá.

(e) **Metódy kompresie** umožňujú klientovi identifikovať všetky kompresie dát, ktoré môže použiť. Kompresné metódy sú dôležitou súčasťou SSL, pretože šifrovanie má významný dopad na efektívnosť akýchkoľvek techník kompresie dát. Z tohto dôvodu je dôležité, aby obe strany používali kompresiu dát pri komunikácií, a tým skomprimovali svoje údaje pred ich zašifrovaním.

2. **Správa Server Hello** príjme správu Klient Hello a reaguje na ňu.

   (a) **Verzia** je prvý krok, kde server ustanovil počiatočne komunikačné podmienky. Server definuje verziu SSL alebo TLS protokolu, ktorá sa bude pri komunikácií používať. V prípade, že to klientovi nevyhovuje, môže komunikáciu opustiť.

   (b) **Náhodne číslo** je v podstate rovnaké ako pri správe Klient Hello. Spolu s číslom klienta je toto číslo dôležité pre kryptografické výpočty.

   (c) **ID relácie** slúži na identifikáciu príslušnej relácie. Dôvodom stanovenia tejto relácie je možnosť odkazovať sa na ňu neskôr.

   (d) **Model šifry** oproti správe Klient Hello určuje presné kryptografické parametre, konkrétne algoritmy a veľkosti kľúčov, ktoré sa budu ďalej používať. Server si vyberie jeden zo šifrovacích balíkov uvedených klientom.

   (e) **Metódy kompresie** server používa na identifikáciu metód kompresie dát zo zoznamu klienta a spomedzi nich si jednu vyberie.

3. **Výmena kľúča serveru** obsahuje informácie o verejnom kľúči. Presný formát informácií o kľúči závisí od použitého algoritmu verejného kľúča. Táto správa sa prenáša bez šifrovania, takže môžu byť zverejnené len informácie o verejnom kľúči. Klient verejný kľúč ďalej použije na šifrovanie.

4. **Server Hello hotovo** oznamuje klientovi, že server ukončil proces. Samotná správa nemá ďalšie informácie, ale je dôležitá pre klienta, pretože môže prejsť do ďalšej fázy komunikácie.

5. **Výmena kľúča klienta** informuje server o kľúčových informáciách klienta.

V prípade, že server dokončil svoju časť ustanovenia parametrov komunikácie, klient takisto posiela správu šifrovanú pomocou verejného kľúča. Toto šifrovanie chráni dôležité informácie v sieti a umožňuje klientovi overiť, či je server právoplatný majiteľ súkromného kľúča. Nikto iný nebude schopný dešifrovať túto správu. Táto operácia je dôležitá na ochranu pred útočníkom, ktorý zachytáva správu zo servera.

6. **Výmena šifry** je okamih, kedy sú obe strany pripravené začať využívať bezpečnostné služby, ktoré si ustanovili.
7. **Ukončenie** je proces, kedy dochádza k uzavretiu komunikácie.

2.4.4 Certifikačné autority (CA)

Vydavateľ verejného kľúča je tradične známy ako **certifikačná autorita** (CA) a zohráva dôležitú úlohu pri vytváraní dôvery medzi komunitou používateľov. Certifikačná autorita digitálne podpisuje všetky certifikáty, potvrdzujúce platnosť verejných kľúčov, ktoré obsahuje. Ak užívatelia dôverujú certifikačnej autorite, dôverujú akémukoľvek certifikátu, ktorý poskytuje.

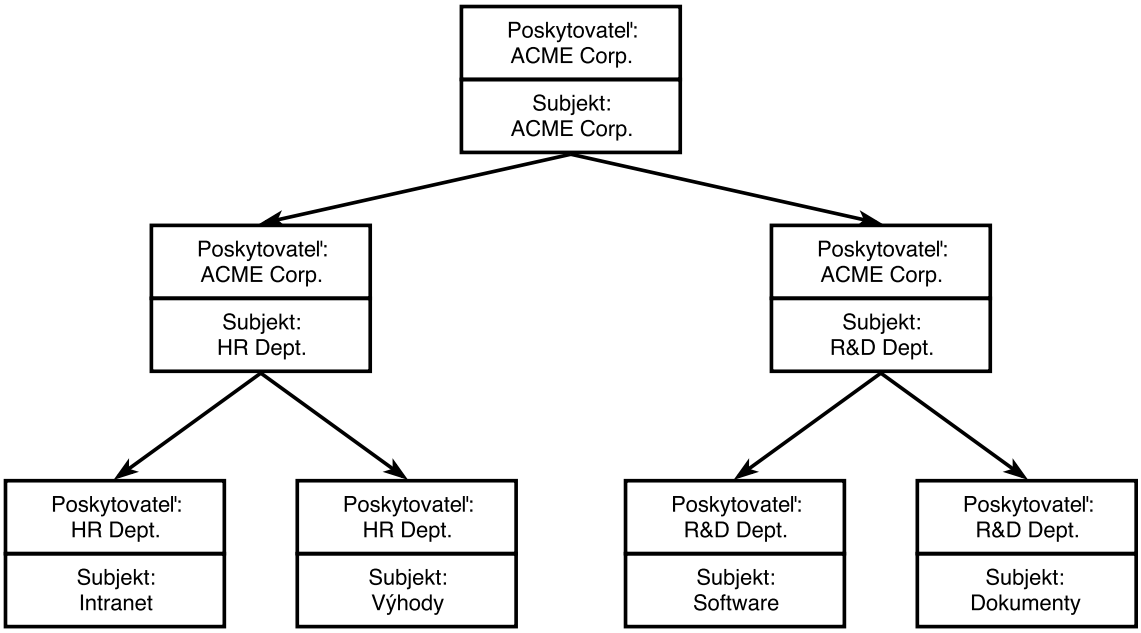
V mnohých prípadoch môže byť certifikačná autorita identifikovaná buď ako súkromná, alebo verejná. **Súkromné** autority používajú organizácie, ktoré poskytujú certifikáty ich vlastným užívateľom. Napríklad spoločnosť môže poskytnúť certifikáty ich vlastným zamestnancom. Spoločnosť by potom mohla vytvoriť svoju internú sieť, aby overila užívateľov pred udelením prístupu k dôležitým údajom. Systémy v rámci tejto siete budú dôverovať tejto certifikačnej autorite, avšak vonkajšie systémy, vrátane verejných serverov, ju budú považovať za nedôveryhodnú.

Internet je však verejná sieť a webová bezpečnosť spočíva na verejných certifikačných autoritách. **Verejná** certifikačná autorita poskytuje certifikáty pre širokú verejnosť a môže osvedčiť totožnosť ako jednotlivcov, tak aj organizácie. Verejnú autoritu môžeme prirovnať k digitálnemu „notárovi“, ktorý potvrdzuje totožnosť príslušného poverenia.

Certifikačné autority sú samy často identifikované ich vlastnými certifikátmi, ale ich certifikáty sa odlišujú od štandardných certifikátov jedným dôležitým aspektom: subjekt a emitent (vydavateľ) sú jedno a to isté. Autorita potvrdzuje svoju vlastnú totožnosť. Toto je podstatný rozdiel od bežných certifikátov. Každá strana, ktorá obdrží bežný certifikát, môže skontrolovať jeho podpis, aby sa rozhodla, či dôveruje verejnému kľúču v tomto certifikáte. Pokiaľ je podpis certifikátu platný a poskytovateľ je dôveryhodný, potom príjemca môže bezpečne dôverovať verejnému kľúču. Na druhej strane, overenie podpisu certifikátu nepomôže zistiť dôveryhodnosť certifikačnej autority. Platnosť certifikátov sa preto musí stanoviť inými metódami.

V prípade bezpečnosti webového obchodu dôveryhodnosť certifikačnej autority

závisí vo veľkej miere od prehliadačov, ktoré by mali rozpoznať verejné certifikáty. Niekedy je zložité pre certifikačnú autoritu dôsledne sledovať všetky strany, ktorých totožnosť overuje. Najmä v prípade, ak počet certifikátov narastá. Našťastie certifikáty podporujú koncept hierarchií, ktoré zlepšujú rozšíriteľnosť jednej monolitickej autority. S použitím tejto hierarchie certifikačná autorita nemusí potvrdzovať všetky identifikačné údaje. Namiesto toho si každá autorita môže určiť svoje dcérske autority. Ako ukazuje obr. 2.7, spoločnosť disponuje hlavnou certifikačnou autoritou a dvomi dcérskymi, jeden pre ľudské zdroje a druhý pre vývoj a výskum.



Obr. 2.7: Hierarchia certifikačných autorít podľa funkcií certifikátov

Obzvlášť významná funkcia hierarchií certifikátov spočíva v tom, že sa nevyžaduje, aby všetky strany automaticky verili všetkým certifikačným autoritám, ktoré využívajú. Jediná autorita, ktorej dôvera musí byť overená v celom systéme, je rodičovská certifikačná autorita [14].

3 E-mailové servery

E-mailový alebo poštový server je aplikácia alebo počítač v sieti, ktorého jediným účelom je pôsobiť ako virtuálna pošta. Server ukladá prichádzajúcu poštu na distribúciu miestnym používateľom a odošle odchádzajúce správy. Tento princíp sa realizuje pomocou modelu aplikácie klient-server na odosielanie a prijímanie správ pomocou *Simple Mail Transfer Protocol* (SMTP), viac obrázok 3.1 .

Samotný proces odosielania a prijímania e-mailov sa vykonáva v niekoľkých krokoch. Ak užívateľ odošle e-mail, jeho adresa je vo forme user@doména.ext. Klient posiela e-mail na server pre odosielanie pošty prostredníctvom protokolu SMTP. Ten kontroluje adresu a určuje, kam bude pošta odoslaná. Problém nastáva pri preklade adresy. Z toho dôvodu musí byť do tohoto procesu zakomponovaný DNS (*Domain Name System*), ktorý preloží adresu domény na IP adresu. Okrem toho zisťuje, či sa na serveri nachádzajú záznamy MX. V tomto momente má SMTP server správne informácie a správa sa odošle z tohoto servera na server pre výmenu pošty cieľovej domény. Tento server je označovaný ako MTA (*Mail Transfer Agent*). Nasleduje proces, ktorý rozhodne akým spôsobom je ideálne odoslať správu. Po prijatí správy si ju klient na strane príjemcu môže prečítať. [16]

Obr. 3.1: Proces odosielania e-mailových správ

3.1 MX záznamy

Záznam o výmene pošty (záznam MX) je typ certifikovaného a overeného záznamu o zdroji v DNS, ktorý určuje poštový server, zodpovedný za prijímanie e-mailových správ v mene domény príjemcu. Taktiež určuje hodnotu preferencie, používanú na uprednostnenie doručovania pošty, ak je k dispozícii viacero mailových serverov. Súbor záznamov MX určuje, ako má byť e-mail smerovaný pomocou protokolu SMTP.

Mechanizmus MX umožňuje spúšťať viaceré poštové servery pre jednu doménu a taktiež správcom určiť poradie, v ktorom by mali byť vyskúšané. Ak doména obsahuje MX záznam s jediným serverom, tak nezáleží na preferencií, a MTA sa pokúsi doručiť poštu na uvedený server. V prípade, ak sa v MX zázname vyskytuje viacero

serverov, preferenčné číslo každého záznamu určuje relatívnu prioritu uvedeného servera. Keď vzdialený klient (zvyčajne iný poštový server) robí MX vyhľadávanie pre názov domény, dostane zoznam serverov a ich preferenčné čísla. Najmenšie číslo predvoľby má najvyššiu prioritu.

Na zabezpečenie spoľahlivého prenosu pošty musí byť klient SMTP schopný vyskúšať (a opakovať) každú z príslušných adries v tomto zozname v poradí.

Servery s nižšou prioritou, napríklad záložné MX alebo sekundárne MX, zvyčajne udržujú správy vo fronte, čakajúc na to, aby bol primárny server dostupný. Záložný MX slúži ako server pre ukladanie a preposielanie správ [17].

3.2 SPF záznamy

Sender Policy Framework (SPF) je jednoduchý systém na overovanie e-mailov, ktorý je schopný detekovať falšovanie hlavičky e-mailu (tzv. *e-mail spoofing*), kedy sa zdá, že správa pochádza od niekoho iného, ako bol skutočný zdroj.

Pridanie záznamu SPF do DNS je najlepší spôsob, ako zastaviť spamerov. Záznam SPF okrem toho zníži počet legitímnych e-mailových správ, ktoré sú označené ako spam alebo vrátené späť poštovými servermi príjemcov.

Záznam SPF nie je 100% efektívny, pretože bohužiaľ nie všetci poskytovatelia pošty ho kontrolujú. Mnohí z nich však môžu zaznamenať výrazné zníženie nežiadúcich správ.

SPF umožňuje vlastníkovi internetovej domény určiť, ktoré počítače sú oprávnené posielať poštu v danej doméne pomocou záznamov DNS.

Prijímače, ktoré overujú informácie SPF v záznamoch TXT, môžu odmietnuť správy z neoprávnených zdrojov predtým, ako prijmú telo správy.

Každá správa SPF má charakteristickú syntax s rôznymi parametrami. Napríklad: „**v=spf1 ip4:192.0.2.0/24 ip4:198.51.100.123 a -all**“. Sú v nej definované tieto mechanizmy:

- **ALL** – zodpovedá všetkým lokálnym a vzdialeným adresám, ktoré nezodpovedajú predchádzajúcim mechanizmom.
- **A** – všetky záznamy A pre doménu sú testované. Ak sa medzi nimi nachádza IP klient, tento mechanizmus sa zhoduje.
- **IP4** – špecifikuje sieťový rozsah IPv4.
- **IP6** – špecifikuje sieťový rozsah IPv6.
- **MX** – všetky záznamy A pre všetky záznamy MX pre doménu sa testujú v poradí priority MX.
- **PTR** – názvy hostiteľov alebo názvy pre IP klientov sa vyhľadávajú pomocou dopytov PTR. Názvy hostiteľov sú potom overené: aspoň jeden zo záznamov A pre názov hostiteľa PTR sa musí zhodovať s pôvodnou IP adresou

klienta. Neplatné názvy hostiteľov sú zamietnuté.

- **EXISTS** – určuje jednu alebo viac domén, ktoré sú zvyčajne označené ako výnimky z definícií SPF.
- **INCLUDE** – odkazuje na pravidlá inej domény. Ak prejdú pravidlá tejto domény, prejde tento mechanizmus. Avšak, ak zistené pravidla zlyhajú, spracovanie pokračuje.

Každý tento mechanizmus môže byť doplnený o 4 kvalifikátory:

- **+ (Pass)** – adresa prešla testom, správa môže byť prijatá. Tento kvalifikátor môže byť vynechaný.
- **? (Neutral)** – nie sú špecifikované žiadne pravidlá.
- **~ (SoftFail)** – adresa neprešla testom, ale výsledok nie je definitívny.
- **- (Fail)** – adresa neprešla testom, všetky e-maily budu odmietnuté.

Okrem toho môžu byť spomínané modifikátory doplnené o 2 rozšírenia:

- **EXP** – ak prijímač SMTP odmietne správu, môže obsahovať vysvetlenie. Vydavateľ SPF môže špecifikovať vysvetľovací reťazec, ktorý odosielatelia uvidia.
- **REDIRECT** – môže byť použitý namiesto mechanizmu ALL na prepojenie na záznam o pravidlách inej domény [18].

4 Návrh a realizácia práce

V tejto kapitole je rozoberaný návrh nástroja, ktorý umožňuje automatizovanú analýzu zabezpečenia webových a e-mailových serverov. Na začiatku je vysvetlená problematika, skúmané mechanizmy a následne samotná realizácia práce.

4.1 Popis riešenia problému

Ako je v teoretickej časti uvedené, existujú protokoly a mechanizmy, ktorá nemusia pokrývať dostatočnú úroveň zabezpečenia. Z toho dôvodu je potrebné vedieť rozlíšiť bezpečné stránky od tých nebezpečných.

V prvom rade sa upriami pozornosť na parametre, ktoré sa budú skúmať u každého webového a e-mailového serveru. Ako už bolo uvedené v prvej kapitole, tak každá stránka môže pri komunikácii s klientom využívať jeden z protokolov HTTP alebo HTTPS. Preto dôkladné preskúmanie použitia komunikačných protokolov je prvotným cieľom k určeniu zabezpečenia komunikácie.

V prípade ak by stránka nepodporovala komunikáciu prostredníctvom zabezpečeného komunikačného protokolu HTTPS, naša správa odosielaná na server nebude šifrovaná. Z toho dôvodu bude takáto komunikácia označená za nezabezpečenú.

Pokiaľ stránka podporuje komunikáciu prostredníctvom HTTPS protokolu, je potrebné ďalej preskúmať všetky mechanizmy, ktoré zabezpečená komunikácia zahŕňa.

1. **Verzia kryptografického protokolu**
   - Dôležitú rolu pri zabezpečenej komunikácií zohráva vhodné použitie verzie kryptografického protokolu SSL alebo TLS. Ako už evolúcia vývoja týchto protokolov naznačuje, staršie verzie už nie sú ďalej považované za bezpečné a dajú sa napadnúť.
   - V prípade, ak by server nepodporoval súčasné verzie protokolov považované za bezpečné, tak sa takáto komunikácia bude ďalej považovať za rizikovú.
2. **Použitie šifrovacej sady**
   - Je to súbor algoritmov, ktoré nám pomáhajú zabezpečiť sieťové pripojenie pomocou protokolu TLS alebo SSL. Zvyčajne obsahuje nasledujúce parametre.
     (a) Algoritmus na výmenu kľúčov, ktorý chráni informácie potrebné pre vytvorenie zdieľaných kľúčov. Tieto algoritmy sú asymetrické a fungujú pre relatívne malé množstvo správ.
     (b) Algoritmus digitálneho podpisu, ktorý analyzuje zvolený tip hashovacieho algoritmu, používaný certifikátom pri digitálnom podpise. Je

to matematická schéma na preukázanie pravosti digitálnych správ alebo dokumentov. Ak by požitý algoritmus nebol dostatočné silný, komunikácia by mohla byť napadnuteľná. Takáto komunikácia by sa vystavovala riziku možnosti prelomenia hashovacieho algoritmu a mohlo by dôjsť k úniku správy.

(c) Hromadnú šifru na výmenu dát, ktorá šifruje správy vymieňané medzi klientmi a serverom. Tieto algoritmy sú symetrické a dobre fungujú pri veľkom množstve údajov.

(d) Algoritmus overovania správ MAC, ktorý sa používa na autentifikáciu správ – inými slovami, na potvrdenie, že správa pochádza od uvedeného odosielateľa.

- Pokiaľ by šifrovacia sada podporovala bezpečnostné mechanizmy, ktoré nespĺňajú normy určujúce dostatočnú úroveň zabezpečenia, bude lokalizovaný problém.

3. **Overenie certifikačnej autority**
   - Certifikačná autorita obsahuje všetky certifikáty, ktoré uľahčujú využívanie PKI (*Public Key Infrastructure*) tým, že svojou autoritou potvrdzujú pravdivosť údajov, ktoré sú uvedené vo verejnom kľúči. Ak by certifikačná autorita nebola dôveryhodná, tak spoločnosť, ktorá vydala certifikát, nie je overená a je dôležité preskúmať vlastnosti takéhoto certifikátu.
   - Ak sa užívateľ rozhodne dôverovať nejakej certifikačnej autorite, mal by vopred vedieť o stave jej dôveryhodnosti.
   - Aplikácia bude zisťovať, či príslušná stránka obsahuje certifikáty, ktoré spadajú pod dôveryhodnú certifikačnú autoritu, a tým osvedčí totožnosť organizácie.
   - Okrem toho každý certifikát obsahuje podpisový algoritmus, ktorý špecifikuje algoritmus verejného kľúča a hashovací algoritmus. Overenie sily týchto algoritmov je ďalší krok k určeniu dôveryhodnosti certifikátov.
4. **Doba platnosti certifikátov**
   - Nakoniec preskúmame platnosť certifikátov. Každý certifikát obsahuje informáciu o tom, kedy bol vydaný a dokedy je v platnosti.

Všetky tieto parametre sa budú testovať na priloženej sade domén, ktoré sa získajú zo súboru vo formáte csv.

Na konci analýzy bude možné zistiť, koľko webových serverov je v skutočnosti zabezpečených. Na druhej strane sa určí počet rizikových serverov, ktoré buď poskytujú nešifrovanú komunikáciu, alebo nespĺňajú dostatočnú úroveň zabezpečenia. Okrem toho budú vyhodnotené štatistiky najpoužívanejších bezpečnostných mechanizmov, ktoré sa podieľajú na zabezpečení komunikácie.

4.2 Zoznámenie sa s nástrojom Docker

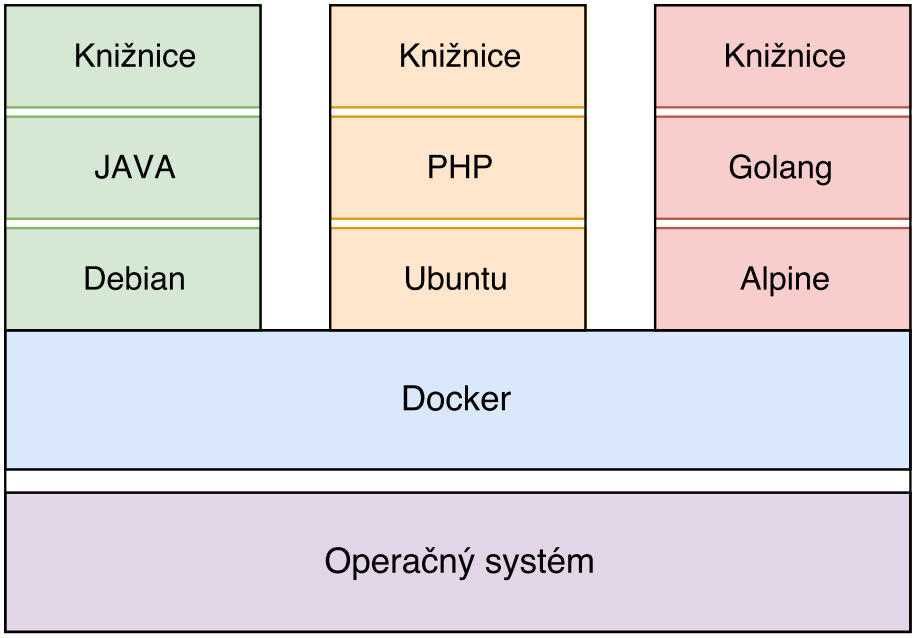
Docker je nástroj určený na uľahčenie vytvárania, nasadzovania a spúšťania aplikácií pomocou kontajnerov. Cieľom je poskytnúť jednotné rozhranie pre aplikácie v **kontajneroch** v prostredí Linuxu aj Windowsu.

Kontajnery umožňujú vývojárovi zabaliť aplikáciu so všetkými potrebnými časťami, ako sú napríklad knižnice a iné potrebné prvky nevyhnutné na spustenie aplikácie (viac obr. 4.1).

Vďaka *Dockeru* sa môže vývojár uistiť, že jeho aplikácia bude fungovať na ľubovoľnom inom počítači.

Docker sa dá čiastočne prirovnať k virtuálnemu stroju. Na rozdiel od virtuálneho počítača však *Docker* namiesto vytvorenia celého virtuálneho operačného systému umožňuje aplikáciam používať to isté jadro ako systém, na ktorom bežia. Tým poskytuje výrazný nárast výkonu a znižuje veľkosť aplikácie.

Jedna z dôležitých vlastností tohoto nástroja je, že je ***open source***. To znamená, že k nemu môže ktokoľvek prispieť a rozšíriť ho tak, aby vyhovoval jeho vlastným potrebám.

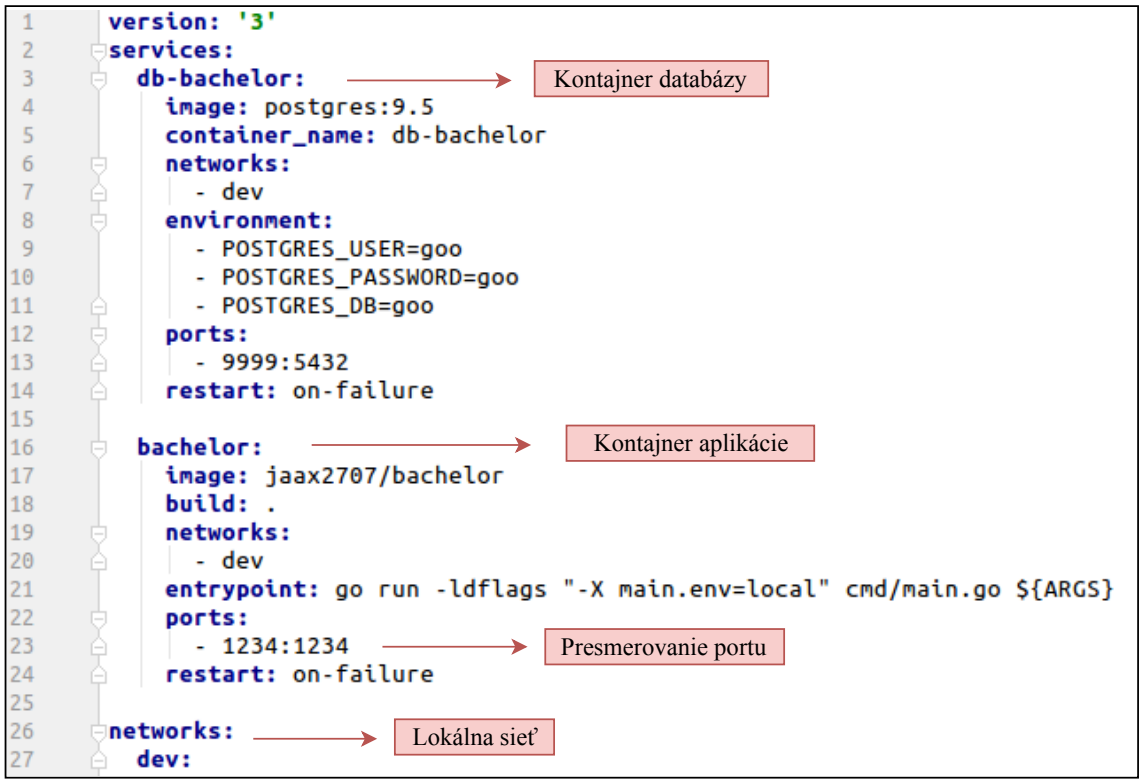


Obr. 4.1: Realizácia kontajnerov v nástroji Docker

Praktická časť bola realizovaná pomocou tohto nástroja, čím sa podstatne zjednodušila možnosť spustiť vytvorenú aplikáciu na inom zariadení.

Keďže v aplikácií sa používajú dva kontajnery, je vhodné použiť pre ich spustenie nástroj ***Compose***, ktorý slúži na používanie multi-kontajnerových aplikácií *Docker*.

Kontajnery sa definujú v konfiguračnom súbore ***docker-compose.yml***. Príklad takéhoto súboru je možné vidieť na obrázku 4.2.



Obr. 4.2: Ukážka súboru docker-compose.yml

Potom pomocou jediného príkazu je užívateľ schopný spustiť všetky služby z danej konfigurácie.

Postupne sa stiahnu obrazy jednotlivých časti, tzv. ***docker images***, ako napríklad obraz pre databázu alebo programovací jazyk Go, v ktorom je spracovaná aplikácia. V aplikácií sa nachádza aj špeciálny súbor ***Dockerfile***, ktorý obsahuje príkazy, pomocou ktorých je zostavený nový obraz.

Obrovská výhoda spočíva v tom, že jednotlive obrazy sú spustiteľné bez toho aby sa v systéme inštalovali, takže je veľmi jednoduché ich neskôr zmazať. Okrem toho *Docker* zabezpečuje, že budú k dispozícií najnovšie verzie obrazov. To je dobrá vec z hľadiska bezpečnosti, pretože sa tým zabezpečí, aby sa nenainštaloval žiadny zraniteľný softvér.

Vo vytvorených kontajneroch sa nachádzajú zdrojové kódy a všetky závislosti, nevyhnutné pre spustenie aplikácie.

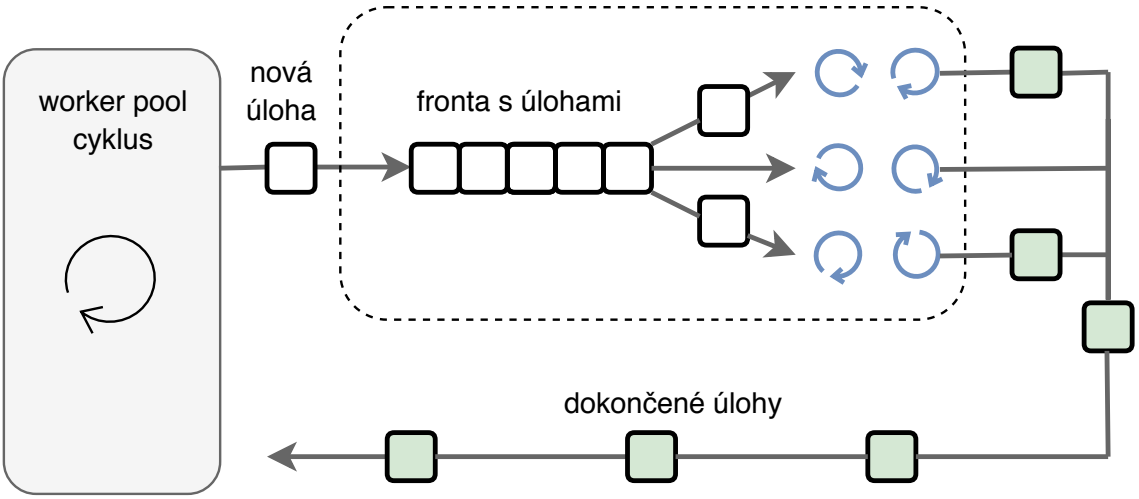
4.3 Použitie softwarového návrhu Worker pool

Jednou z najsilnejších stránok programovacieho jazyka Go je vstavaná súbežnosť a vytváranie komplexne súbežných tokov.

Worker pool je návrhový vzor na dosiahnutie súbežnosti vykonávania procesov v počítačovom programe. V praxi to znamená, že ak by sme chceli vykonať nejakú zložitú operáciu náročnú na čas, tento model nám umožní si ju rozdeliť na viacero menších operácií, ktoré sa budú vykonávať súbežne.

Pred samotným vysvetlením princípu tohoto modelu si vysvetlíme nový pojem ***go routine***. Je to funkcia, ktorá je schopná bežať súčasne s inými funkciami. Go-rutíny možno považovať za špeciálne vlákno s veľmi nízkymi nárokmi na vytvorenie v porovnaní s klasickým vláknom. Preto je možné pre aplikácie vytvorené v programovacom jazyku Go, aby sa súčasne prevádzkovali tisíce *go routines*.

Viacero *go routine* môže čítať z jedného kanála a distribuovať množstvo práce medzi jadrami procesora. V Go je tento model ľahko implementovateľný – stačí spustiť niekoľko *go routines* s kanálom ako parametrom a odoslať hodnoty do tohto kanála, pričom následná distribúcia a multiplexovanie sa vykoná v Go *runtime*. Na obrázku 4.3 je názorná ukážka realizácie tohoto modelu.



Obr. 4.3: Realizácia modelu worker pool

Samotný model funguje na zásade, v ktorej pevný počet m *go routine* (implementovaných v programovacom jazyku Go) vykonáva postupne n úloh.

Napríklad pri testovaní veľkého objemu dát sa vhodne zvolí počet *go routine*, ktoré budu postupne vykonávať tieto úlohy. Keďže pripájanie na webové servery je závisle hlavne od rýchlosti internetu, tento model musí byť prispôsobený dostatočne na to, aby pripojenie stačilo na súbežné napájanie sa na domény.

V prípade, ak by sa nesprávne nastavil počet *go routine*, mohlo by dôjsť k zahlteniu pripojenia a proces by sa značne spomalil.

Keďže pri analýze je vhodné využiť čo najväčšie množstvo dát, výber tohoto návrhového vzoru je výhodný spôsob, akým docieliť dobrý pomer rýchlosti spracovania informácií a výkonu. Ukážku implementácie kódu modelu *worker pool* je možné vidieť na výpise 4.1

Výpis 4.1: Príklad implementacie modelu worker pool v jazyku Go.

```
// WorkerPoolConn creates worker pool
func (w *WebPool) WorkerPoolConn(urls []string) []models.HttpServer {
    startTime := time.Now()

    jobs := make(chan models.Job)
    results := make(chan models.HttpServer)

    go w.Allocate(urls, jobs)

    go w.Result(results)

    w.CreateWorkerPool(environment.NO_OF_WORKERS, jobs, results)

    endTime := time.Now().Sub(startTime).Seconds()

    log.Println("total time taken ", endTime, "seconds")

    return webServers
}
```

4.4 Spracovanie výsledkov analýzy

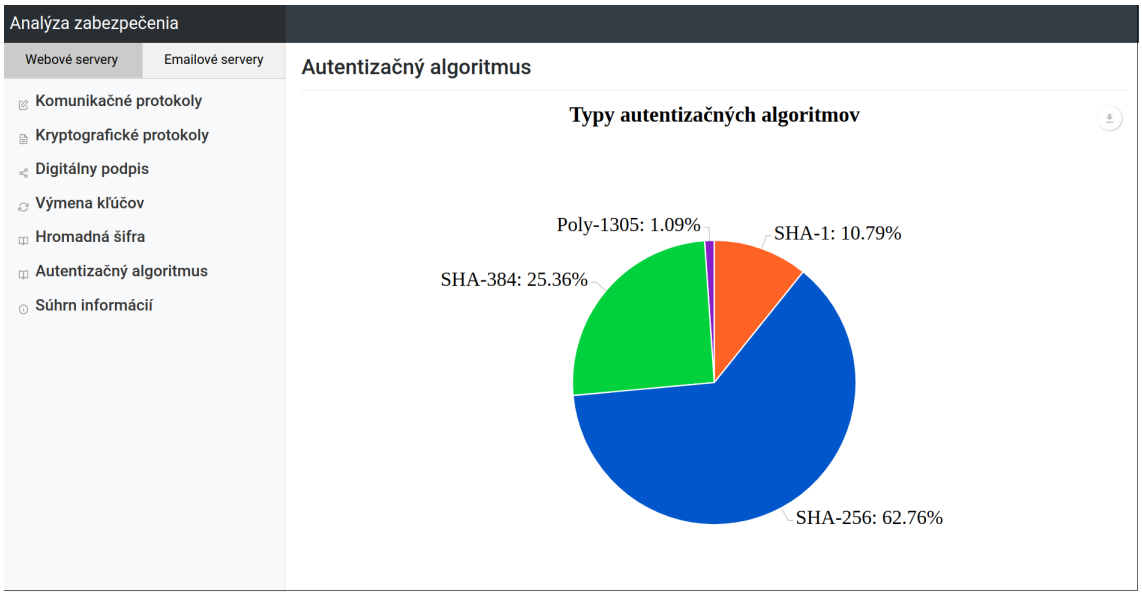
Výsledky analýzy boli sprostredkované za pomoci naprogramovanej aplikácie. Na výstupe boli zhrnuté bezpečnostné mechanizmy serverov využívajúce rôzne typy zabezpečenia. Postup spracovania výsledkov môžeme rozdeliť do 4 krokov:

1. **Parsovanie súboru** vo formáte csv, v ktorom sa nachádza sada domén. K dispozícii boli 3 sady domén. Prvá obsahovala názvy webových serverov z rôznych častí sveta, konkrétne obsahuje až 1 milión domén. Druhá sada obsahuje názvy e-mailových poskytovateľov. A nakoniec bola manuálne vytvorená tretia sada, ktorá pozostávala z prvej sady a obsahovala len cz a sk domény.
2. **Asynchrónne pripájanie** na webové a e-mailové servery pomocou návrhového vzoru *worker pool*. Pri webových serveroch sa nadväzuje spojenie na port 443 (TLS port), pri e-mailových serveroch sa používa port 587. Po úspešnom

pripojení sa získavajú informácie o kryptografických protokoloch, ktorými disponuje daný server. Servery, ktoré odmietli pripojenie na TLS port, boli označené za servery nepodporujúce protokol HTTPS. V prípade, ak by niektoré servery neexistovali alebo doba pripájania (tzv. *Timeout*) prekročí stanovený limit, daný server sa preskočí a pokračuje sa na ďalší.

3. **Analýza serverov**, ktorá pozostáva zo spracovania informácií o kryptografických protokoloch a certifikátoch, ktoré daný server používa. Určuje sa sila jednotlivých algoritmov, ktoré zohrávajú úlohu pri celom procese komunikácie medzi klientom a serverom. Spôsob určenia sily a bezpečnosti týchto mechanizmov je špecifikovaný v RFC 5246 [19]. Po úspešnej analýze sa spracované informácie uložia do databázy (bola použitá databáza *PostgreSQL*).
4. **Zobrazenie výsledkov**. Bol vytvorený lokálny server s dvomi *routami*, pričom jedna slúži na renderovanie HTML šablony a druhá zobrazuje štatistiky v grafoch za pomoci *JavaScriptu*. Užívateľ si môže zvoliť, či chce zobraziť štatistiky webových alebo e-mailových serverov.

Na obrázku 4.4 je zobrazená HTML šablona, ktorá je po analýze vyrenderovaná na lokálny server.



Obr. 4.4: Výsledky analýzy zabezpečenia webových serverov

V ľavej časti sa nachádzajú dva taby, ktoré rozdeľujú analýzu do dvoch kategórií: webové a e-mailové servery. V pravej časti sa nachádza priestor, v ktorom sa zobrazujú grafy s vyhodnotenou analýzou.

4.5 Návod na spustenie aplikácie

V tejto časti je rozobraný spôsob, akým aplikáciu spustiť a škálovať podľa vlastných potrieb. Aplikácia sa spúšťa pomocou *Makefilu*, v ktorom sú definované pravidlá, podľa ktorých sa pristupuje k určitým typom súborov. Okrem toho je možné nastaviť limit, ktorý definuje počet serverov, ktoré budú do analýzy zakomponované. V *Makefile* sú definované dva príkazy: ***build*** a ***run***.

Build poskytuje inicializáciu kontajneru, v ktorom sa nachádza zdrojový kód aplikácie. Okrem toho spustí inštaláciu balíčkov, potrebných pre spustenie aplikácie.

```
$ make build
```

Príkaz *run* aplikované zmeny spracuje a zahájí spustenie aplikácie. Za príkaz run je potrebné definovať argumenty, ktoré špecifikujú typ analýzy a limit (počet serverov, ktoré budu testované). Ukážka príkazu *run* môže vyzerať nasledovne:

```
$ make run ARGS="web 1000"
```

V tomto prípade by to znamenalo, že bude spustená analýza webových serverov s nastavením limitu na 1000.

Ďalším argumentom, ktorý si užívateľ môže zvoliť, je analýza e-mailových serverov:

```
$ make run ARGS="email 100"
```

V prípade spustenia analýzy pre oba typy serverov súčasne sa zvolí argument all. Ak užívateľ nenastaví limit počtu testovaných serverov, budú použité východzie hodnoty definované v aplikácii.

```
$ make run ARGS="all"
```

V prípade, ak by užívateľ chcel nahliadnuť do manuálu implementovaného CLI (*Command-line interface*), spustí *Makefile* s argumentom *help*.

```
$ make run ARGS="--help"
```

Následne sa zobrazí celá dokumentácia k spusteniu aplikácie.

5 Rozbor výsledkov

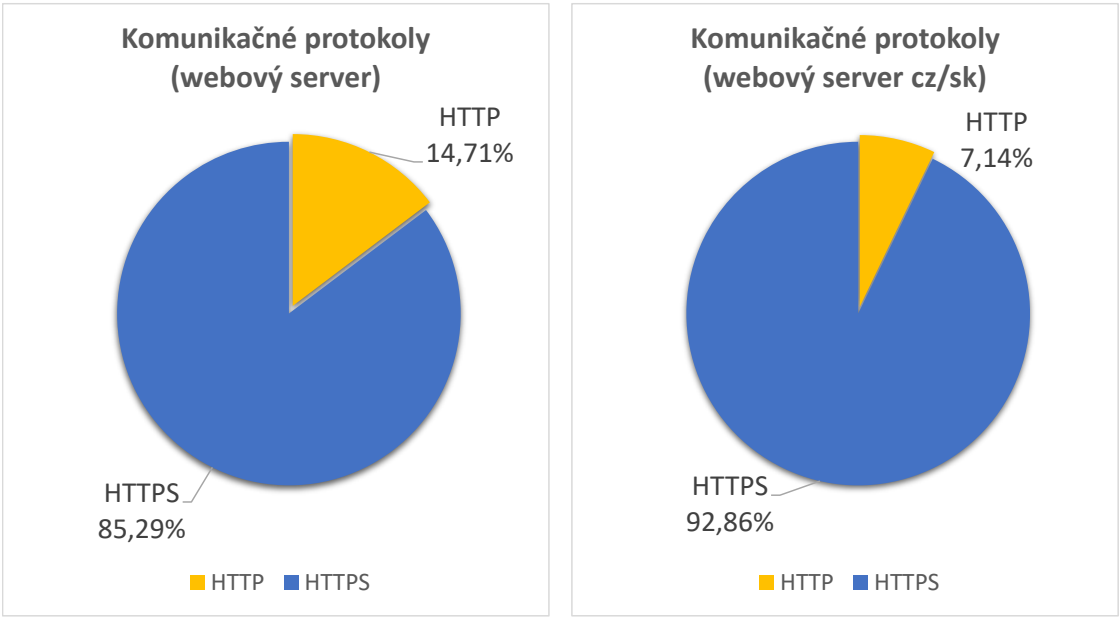
V tejto kapitole sa postupne rozoberú výsledky skúmania všetkých mechanizmov, ktorými disponujú webové a e-mailové servery.

5.1 Použitie komunikačných protokolov

Na aplikačnej vrstve OSI modelu sú definované komunikačné protokoly HTTP a HTTPS, ktoré sa starajú o prenos informácií. Aplikácia poskytuje identifikáciu tohoto protokolu pre každú doménu, ktorá sa nachádza v testovacej sade.

V prípade, ak server odmietne pripojenie na TLS port – a to z dôvodu, že takúto komunikáciu nepodporuje, prenos informácií sa uskutočňuje prostredníctvom protokolu HTTP.

Na obrázku 5.1 môžeme vidieť štatistiky použitia komunikačných protokolov pre univerzálnu sadu domén a špeciálnu sadu domén, ktorá obsahuje len sk a cz domény.



Obr. 5.1: Komunikačné protokoly HTTP a HTTPS

Z výsledkov je možné vyvodiť záver, že v dnešnej dobe už väčšina serverov podporuje protokol HTTPS. Z celkového počtu analyzovaných serverov (84 273) používalo šifrovanú komunikáciu 85 % webových serverov.

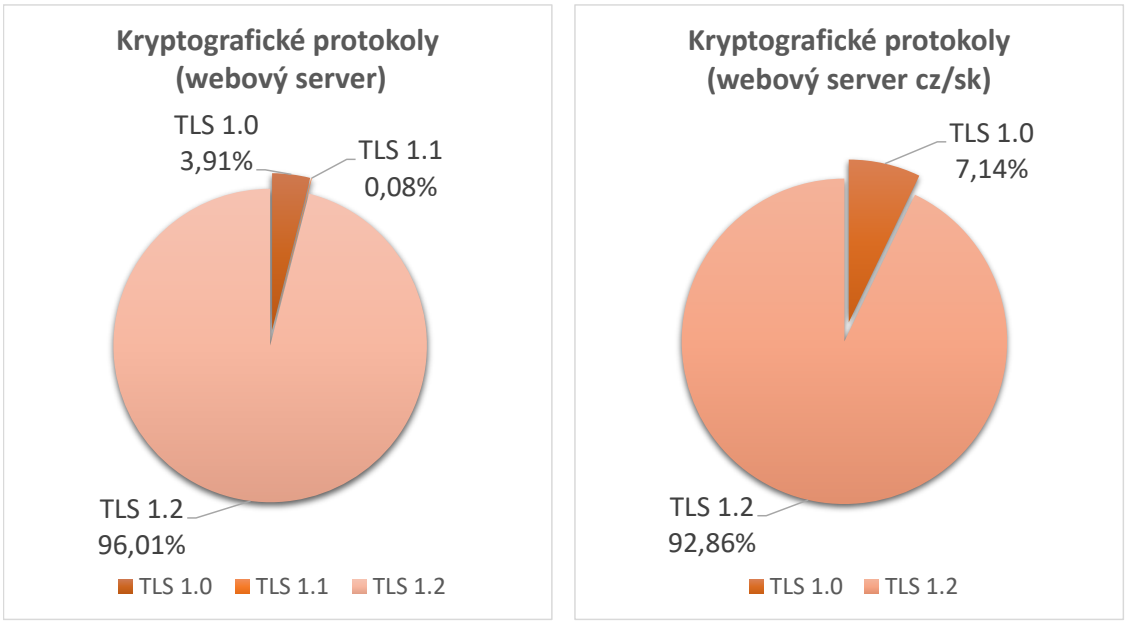
V prípade českých a slovenských domén bolo preskúmaných 5 852 serverov a až 92 % používa protokol HTTPS.

Pri analýze e-mailových serverov bolo zistené, že 86 % z nich podporuje HTTPS protokol.

5.2 Verzie kryptografických protokolov

Okrem protokolov HTTP a HTTPS sa na aplikačnej vrstve nachádzajú aj kryptografické protokoly SSL a TLS, ktoré zaisťujú bezpečný prenos informácií. V prípade, ak server odpovie, že podporuje tento typ zabezpečenia, budú následne preskúmané verzie týchto protokolov.

Najnovšia verzia protokolu je TLS 1.2, ktorá zatiaľ nezaznamenala väčšie nedostatky. Na obrázku 5.2 sú uvedené použité verzie kryptografických protokolov v porovnaní s českými a slovenskými doménami.



Obr. 5.2: Kryptografické protokoly SSL a TLS

Z výsledkov vyplýva, že z analýzy univerzálnej sady domén až 96 % serverov podporuje najnovší TLS 1.2 protokol, a len minimum serverov používa staršie verzie protokolu TLS.

Nakoniec sa zistilo, že protokol SSL 3.0 a jeho staršie verzie sa už v dnešnej dobe moc nepoužívajú, a to aj z dôvodu, že nie sú spoľahlivé a boli v nich detekované veľké nedostatky.

V prípade cz/sk domén je vidieť, že až 92 % serverov využíva kryptografický protokol verzie TLS 1.2.

Následne pri analýze e-mailových serverov možno konštatovať, že TLS 1.2 protokol podporuje 85 % serverov, čo je v porovnaní s webovými servermi o niečo menej.

5.3 Analýza šifrovacej sady

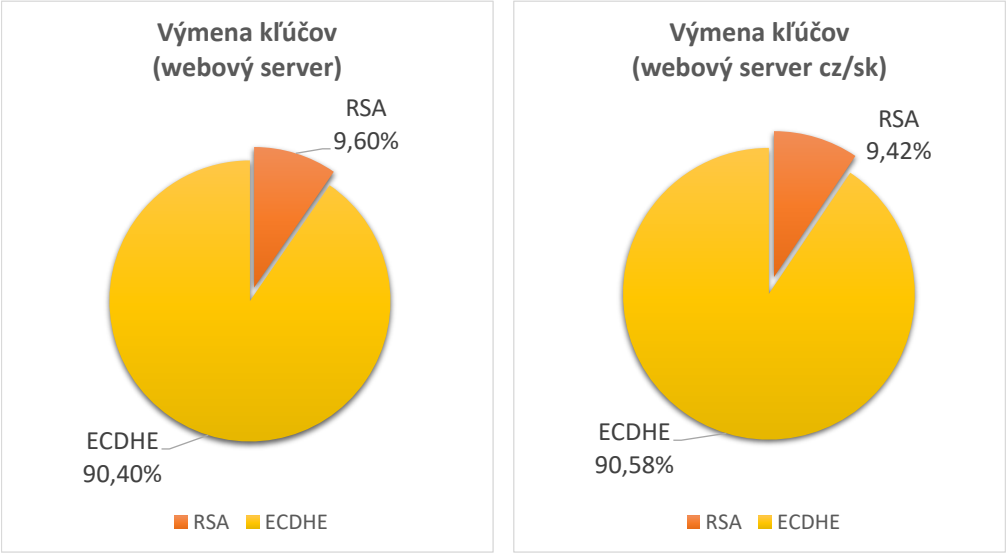
Šifrovacia sada je súbor algoritmov, ktoré pomáhajú zabezpečiť sieťové pripojenie, ktoré využíva kryptografické protokoly TLS alebo SSL.

Súbor algoritmov, ktoré zvyčajne obsahujú šifrovacie sadu, zahŕňa algoritmus výmeny kľúčov, digitálny podpis, hromadný šifrovací algoritmus a autentifikačný algoritmus MAC.

5.3.1 Výmena kľúčov

Algoritmus výmeny kľúčov umožňuje pri komunikácii cez nezabezpečený kanál vytvoriť medzi komunikujúcimi stranami šifrované spojenie. Na konci takéhoto spojenia je vytvorený symetrický šifrovací kľúč, ktorý bude slúžiť pre šifrovanie zvyšku komunikácie.

Medzi najznámejšie algoritmy tohoto procesu patrí **RSA** a *Diffie-Hellman*, konkrétne **ECDHE** (Elliptic Curve Diffie-Hellman Ephemeral).



Obr. 5.3: Algoritmy na výmenu kľúčov

Zistilo sa, že ako najčastejší algoritmus sa pre výmenu kľúčov používa ECDHE, kedy až 90 % serverov uprednostňuje tento mechanizmus pred RSA. V porovnaní s českými a slovenskými doménami je percentuálne zastúpenie tohto typu algoritmu veľmi podobné. Pri e-mailových serveroch mechanizmus ECDHE používalo 74 % serverov.

Výpočtová náročnosť mechanizmu ECDHE je veľmi rýchla, avšak neumožňuje autentifikáciu účastníkov komunikácie, a z toho dôvodu sa používa v kombinácií s inými metódami.

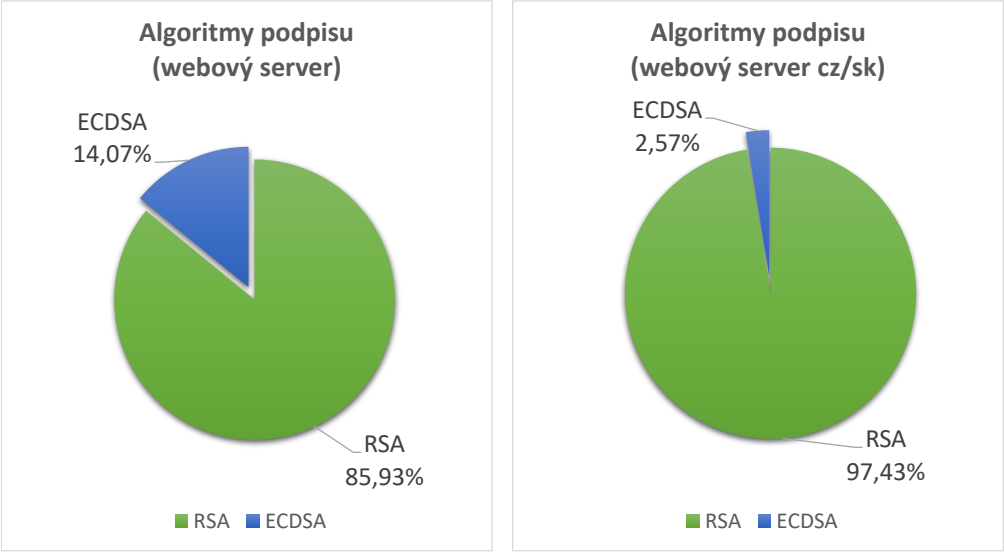
5.3.2 Algoritmus digitálneho podpisu

Je to matematická schéma na preukázanie pravosti digitálnych správ, ktorá využíva asymetrickú kryptografiu. Poskytuje pri komunikácii autentifikáciu, integritu a časové obmedzenie, pokiaľ ma daná relácia platnosť. Samotná autentifikácia sa realizuje v kombinácií s digitálnym certifikátom, ktorý znemožňuje neoprávnenej osobe správu podvrhnúť.

Najznámejšie šifrovacie algoritmy pre digitálny podpis sú RSA a ECDSA. Ak by sme porovnali RSA a ECDSA v približne rovnakej bezpečnostnej úrovni, tak ECDSA je o niečo silnejšia. Avšak dôvodom výberu algoritmu RSA oproti ECDSA pre väčšinu serverov je hlavne rýchlosť overovania.

Z detailnejšieho rozboru algoritmu RSA 2048 a ECDSA 256-bit vyplýva, že ECDSA je o niečo silnejší algoritmus. Keď sa ale pozrieme na rýchlostné testy podľa oficiálnych zdrojov [20], RSA algoritmus zaberie len 0.16 msec oproti ECDSA, ktorý by trval 8.53 msec.

Na obr. 5.4 môžeme vidieť výsledky analýzy.



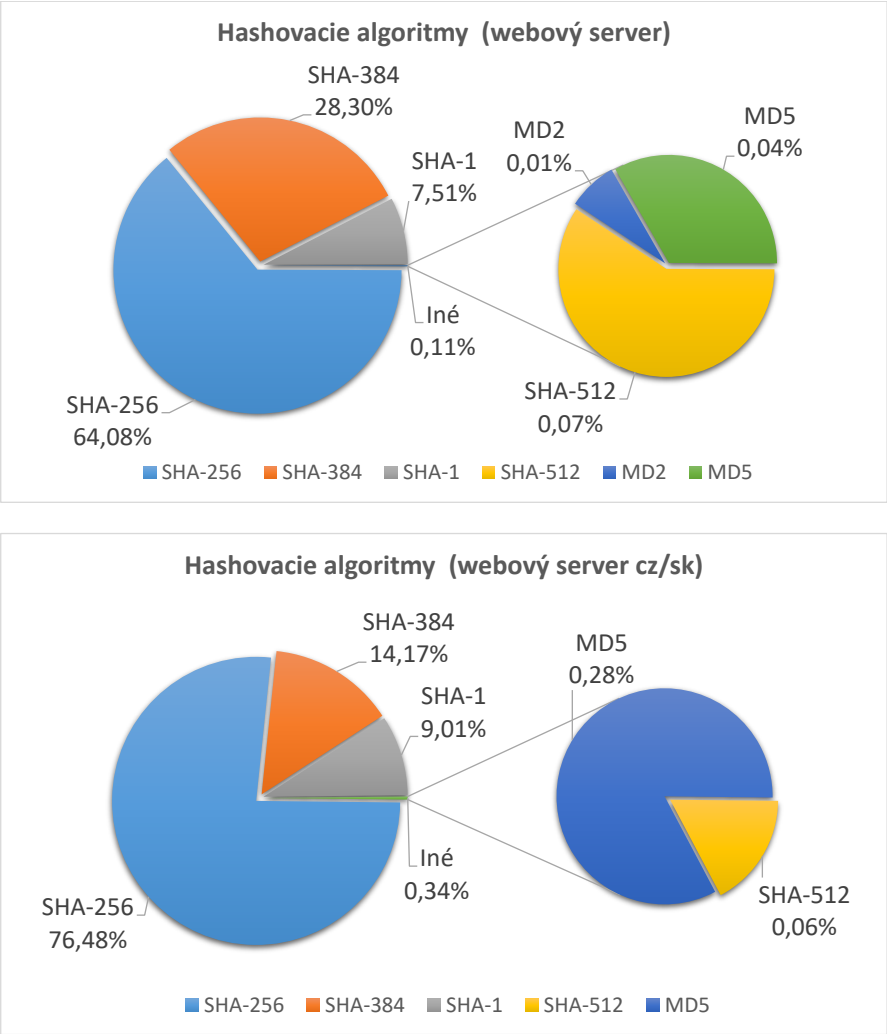
Obr. 5.4: Algoritmy podpisu

Najpoužívanejší podpisový algoritmus je RSA. Až 85 % serverov zvolilo tento typ algoritmu ako svoj ochranný mechanizmus. V súčasnosti neexistuje žiadny dôvod súvisiaci s bezpečnosťou, ktorý by uprednostňoval jeden typ mechanizmu pred akýmkoľvek iným, za predpokladu dostatočne veľkých kľúčov (2048 bitov pre RSA alebo 256 bitov pre ECDSA). Podpora ECDSA je novšia, a preto väčšina serverov využíva, práve mechanizmus RSA.

Pri e-mailových serveroch sa zistilo, že všetky testované servery uprednostňujú mechanizmus RSA.

Digitálny podpis v sebe zahŕňa okrem šifrovacích algoritmov aj hashovacie algoritmy, ktoré zabezpečujú integritu správy.

Obrázok 5.5 zobrazuje použitie všetky typy hashovacích algoritmov, ktoré boli zistené pri analýze.



Obr. 5.5: Hashovacie algoritmy

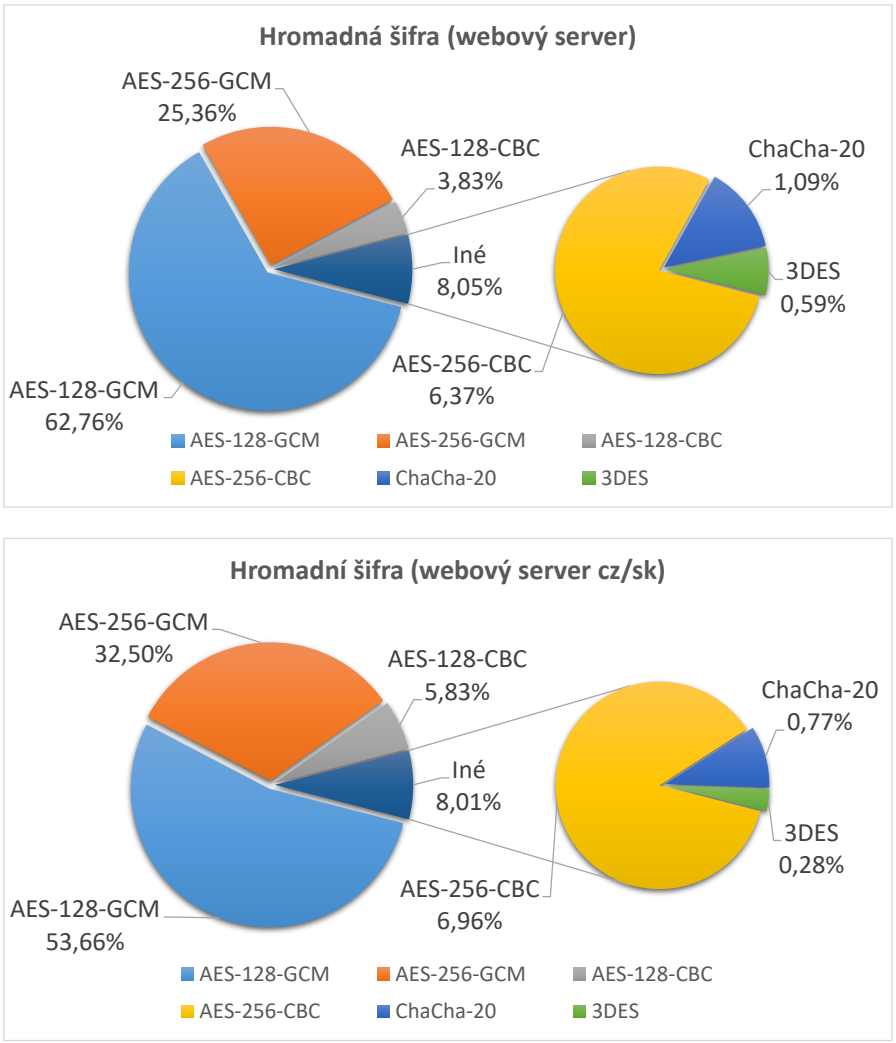
Najpoužívanejším algoritmom je SHA-256, ktorý v porovnaní s jeho novšími verziami poskytuje rovnakú úroveň bezpečnosti. V súčastnosti neexistuje výhoda SHA-384 alebo SHA-512 pred algoritmom SHA-256. Naopak, nevýhodou by mohla byť kompatibilita so systémami, ktoré ich nepodporujú. Algoritmus SHA-256 vyžaduje menej pamäte na ukladanie či prenos a menej výpočtovej výkonnosti.

Na druhej strane boli nájdené servery, ktoré využívajú slabšie algoritmy, ako sú SHA-1 alebo MD-5.

E-mailové servery podobne ako webové preferujú algoritmus SHA-256 pred ostatnými typmi.

5.3.3 Hromadná šifra

Hromadná šifra je symetrický algoritmus, ktorý sa používa na šifrovanie a dešifrovanie veľkého množstva údajov. Na obr. 5.6 môžeme vidieť výsledky analýzy.



Obr. 5.6: Hromadné šifry

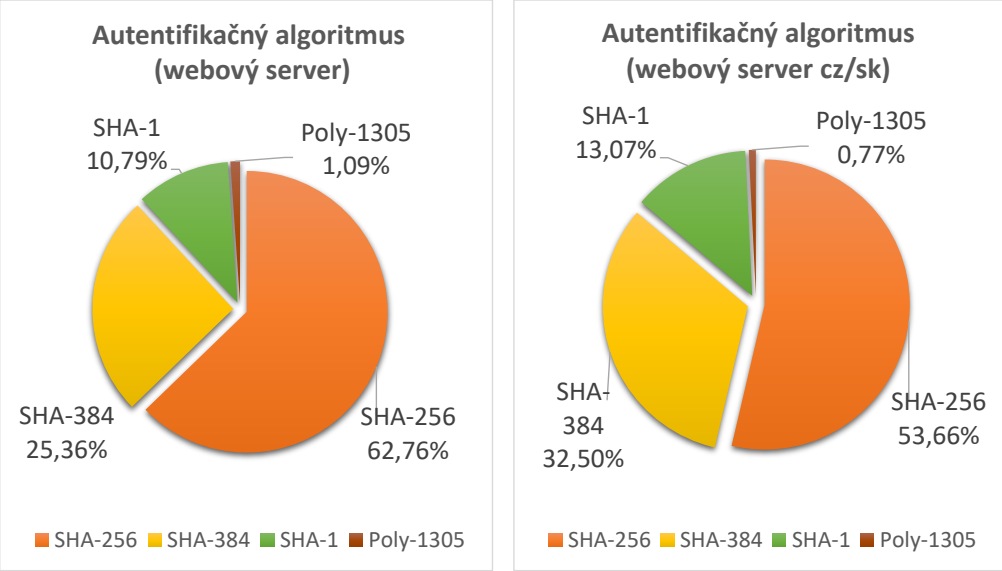
Z analýzy vyplýva, že najpoužívanejším algoritmom pre šifrovanie je AES. Môže nadobúdať rôzne veľkosti, pričom sa zistilo, že najčastejšie sa využíva 128 bitový štandard. Z hľadiska bezpečnosti boli všetky typy veľkosti navrhnuté tak, aby poskytovali dostatočné zabezpečenie. AES sa vyskytuje v 2 rôznych režimoch, GCM alebo CBC. Výhoda režimu GCM spočíva v tom, že poskytuje autentifikáciu údajov.

Okrem toho sa zistilo, že niektoré servery využívajú naďalej algoritmus 3DES, ktorý je síce podľa organizácie *NIST* považovaný za zabezpečený, ale v súčastnosti bol nahradený algoritmom AES.

5.3.4 Autentifikačný algoritmus

Kód na overenie správ MAC zabezpečuje ochranu proti falšovaniu správ. Overuje, že správa pochádza od uvedeného odosielateľa a nebola zmenená. Kým funkcie MAC sú podobné kryptografickým hashovacím funkciám, majú rôzne bezpečnostné požiadavky.

MAC sa líšia od digitálnych podpisov, pretože hodnoty MAC sú generované a overené pomocou rovnakého tajného kľúča. Znamená to, že odosielateľ a príjemca správy sa musia dohodnúť na tom istom kľúči pred začatím komunikácie, ako je to v prípade symetrického šifrovania. Na obrázku 5.7 je zobrazená štatistika použitia jednotlivých autentifikačných algoritmov.



Obr. 5.7: Autentifikačné algoritmy

Z výsledkov môžeme vyvodiť záver, že najčastejším autentifikačným algoritmom je SHA-256, kde viac ako polovica z testovaných serverov využíva konkrétne tento typ algoritmu. Mechanizmus SHA-256 a jeho novšie verzie poskytujú natoľko silný hash, že v súčasnosti neexistuje možnosť ich prekonať.

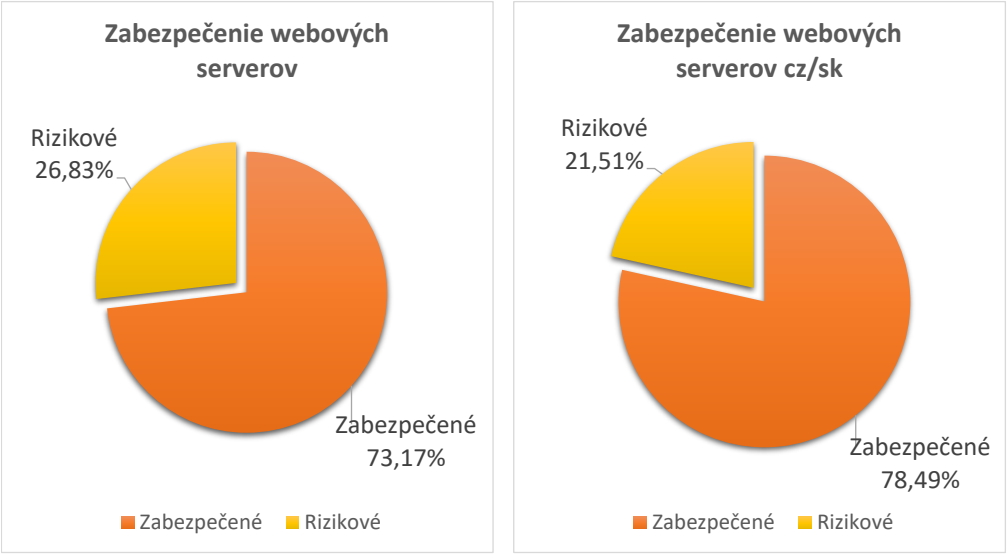
Ďalej sa zistilo, že len 10 % webových serverov využíva algoritmus SHA-1, ktorý bol v roku 2005 prelomený a väčšina serverov tento typ algoritmu časom prestala používať.

V porovnaní s cz/sk doménami je použitie týchto mechanizmov podobné, pričom SHA-1 algoritmus využíva 13 % serverov. Čo sa týka e-mailových serverov, výsledky boli obdobné.

5.4 Súhrn informácií

V tejto časti sa zjednotia jednotlivé analýzy do jedného celku a vyvodí sa záver, v ktorom sa určí počet serverov, ktoré spĺňajú všetky bezpečnostné normy. Tieto normy boli špecifikované na základe možných útokov, ktoré by reálne mohli nastať v prípade, ak by útočník odhalil chybu v zabezpečení.

Medzi skúmané prvky patria komunikačné protokoly, šifrovacia sada a algoritmus digitálneho podpisu. Na obr. 5.8 je zobrazená štatistika zabezpečenia webových serverov.



Obr. 5.8: Zabezpečenie webových serverov

Z celkového počtu 84 273 analyzovaných webových serverov bolo vyhodnotených 73 % ako servery zabezpečené.

Medzi servery, ktoré boli označené za rizikové, patria servery, ktoré podporujú nešifrovanú komunikáciu prostredníctvom protokolu HTTP, a servery, ktoré používajú TLS, ale ich bezpečnostné mechanizmy nie sú dostatočné silné. Z toho dôvodu pri komunikácií môžu nastať rôzne incidenty, či už by sa jednalo o spreneverenie údajov, stratu integrity a ďalšie iné narušenia komunikácie.

V porovnaní s cz/sk doménami sú tie naše o niečo bezpečnejšie, pričom až 78 % webových serverov poskytuje dostatočnú úroveň zabezpečenia.

Ďalej sa zistilo, že niektoré webové servery poskytovali certifikáty, ktoré boli expirované a neboli naďalej platné. Komunikácia síce naďalej zostane šifrovaná, ale užívatelia dostanú škaredé varovné signály o bezpečnosti príslušnej stránky. Väčšina ľudí sa na základe toho rozhodne nepokračovať v práci na tejto stránke a opustí ju.

6 Záver

Bakalárska práca sa venovala problematike zabezpečenia webových a e-mailových serverov. Táto téma je v dnešnej dobe aktuálna a venuje sa jej čoraz väčšia pozornosť, pretože pokrok v oblasti informatiky neustále napreduje, a tak je nevyhnutné zlepšovať bezpečnostné mechanizmy.

Na začiatku práce sú popisované komunikačné protokoly, ich použitie a základný rozdiel medzi nimi. Ďalej sú vysvetlené kryptografické protokoly, ktoré zabezpečujú šifrované komunikačné spojenie medzi klientom a serverom. Nakoniec je klasifikovaná architektúra zabezpečenia serverov, pričom sú rozobraté mechanizmy dôležité pre určenie úrovne bezpečnosti jednotlivých serverov.

V prvom kroku riešenia problému bolo potrebné definovať bezpečnostné prvky, ktoré ovplyvňujú silu zabezpečenia webových a e-mailových serverov. Konkrétne sa jedná o použitie komunikačných protokolov, verzie kryptografických protokolov, rozboru šifrovacej sady a certifikátov. Najdôležitejším prvkom bolo použitie kryptografických protokolov, ktoré zabezpečujú šifrovanie komunikácie. Následne sa bral ohľad na silu jednotlivých algoritmov, ktorými server disponoval, a ich celkový dopad na bezpečnosť komunikácie.

Druhým potrebným krokom bola optimalizácia aplikácie. Keďže analýza sa vykonáva pre väčší počet webových serverov, bolo potrebné vybrať vhodný postup spracovania jednotlivých pripojení. K tomu bol zvolený návrhový vzor *Worker pool*, ktorý poskytuje asynchrónne pripájanie na webové servery. Efektívne sa zvolí počet vlákien, v ktorých sa budú vykonávať požadované procesy.

Samotná analýza bola aplikovaná na tri rôzne sady domén. Prvá, univerzálna sada domén, obsahovala celosvetové webové servery. Druhá sada pozostávala zo súboru špeciálne vytvoreného pre české a slovenské národné domény. V tretej sade sa nachádzali názvy e-mailových poskytovateľov.

Praktickým výstupom práce je aplikácia, vytvorená v programovacom jazyku Go, ktorá automatizovane analyzuje zabezpečenie webových a e-mailových serverov. Spustenie projektu je realizované pomocou nástroja *Docker*, ktorý zabezpečuje funkčnosť aplikácie na všetkých zariadeniach, bez nutnosti manuálnej inštalácie súčastí, na ktorých je aplikácia závislá. Užívateľ tak nemusí riešiť inštaláciu programovacieho jazyka Go, ako aj databázy, ktorá je tak isto implementovaná v platforme *Docker*. Pre spustenie aplikácie je potrebná inštalácia *Dockeru* do systému, v ktorom chce užívateľ aplikáciu spustiť.

Po skončení analýzy sú zobrazené výsledky do HTML šablony, ktorá je *renderovaná* na lokálny server. Užívateľ ma následne možnosť si vybrať a zobraziť štatistiky jednotlivých bezpečnostných prvkov. Výsledky sú spracované a zobrazené na server vo forme grafov.

Z výsledkov analýzy vyplýva, že v súčasnosti väčšina serverov využíva šifrovanú komunikáciu. Na druhej strane bolo nájdených 14 % domén, ktoré poskytujú komunikáciu formou obyčajného textu a z toho dôvodu môže dôjsť k strate integrity a sprenevereniu dát. V prípade českých a slovenských národných domén bolo zistené, že aktuálne disponujú vysokou úrovňou zabezpečenia, pričom až 92 % serverov používalo šifrovanú komunikáciu.

Ďalej boli analyzované všetky prvky šifrovacej sady, ktorú podporoval daný server. Ako prvý mechanizmus, ktorý poskytuje kryptografický zabezpečené komunikačné spojenie, je výmena kľúčov. Najpoužívanejším algoritmom pre tento proces bola eliptická krivka *Diffie-Hellman* (ECDHE), kde až 90 % serverov si zvolilo tento typ mechanizmu.

Samotný proces výmeny kľúčov neposkytuje autentifikáciu. Z toho dôvodu servery používajú digitálny podpis, overený certifikačnou autoritou. Z analýzy bolo zistené, že pre digitálny podpis servery najčastejšie volia podpisový algoritmus RSA v kombinácií s hashovacím algoritmom SHA-256.

Ďalším skúmaným mechanizmom bola hromadná šifra, ktorá poskytuje vysoký stupeň bezpečnostného maskovania a znemožňuje narušiteľovi systému odhaliť legitímnu činnosť s akoukoľvek mierou presnosti. Z analýzy bolo zistené, že vo veľkej miere sa používa symetrická bloková šifra AES.

Posledným prvkom šifrovacej sady je autentifikačný algoritmus (MAC), ktorý zabezpečuje ochranu proti falšovaniu správ. Po preskúmaní týchto algoritmov bolo zistené, že 10 % webových serverov používalo kryptografickú hashovaciu funkciu SHA-1, ktorá je od roku 2005 prelomená.

Nakoniec boli zjednotené jednotlivé analýzy bezpečnostných mechanizmov a určený počet serverov, ktoré poskytujú spoľahlivé komunikačné spojenie. Po skončení analýzy webových serveroch bolo zistené, že 73 % z nich je zabezpečených. Pri českých a slovenských doménach výsledky boli o niečo lepšie, pričom až 78 % serverov splnilo bezpečnostné požiadavky.

Literatúra

[1] Communication Protocol. *Techopedia* [online]. [cit. 27. 10. 2017]. Dostupné z: https://www.techopedia.com/definition/25705/communication-protocol

[2] An overview of HTTP. *MDM web docs* [online]. 2017 [cit. 27. 10. 2017]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[3] Hypertext Transport Protocol Secure (HTTPS). *Techopedia* [online]. [cit. 27. 10. 2017]. Dostupné z: https://www.techopedia.com/definition/5361/hypertext-transport-protocol-secure-https

[4] Hodges, Jackson a Barth, HTTP Strict Transport Security (HSTS). *Internet Engineering Task Force (IETF)* [online]. 2012 [cit. 30. 10. 2017]. Dostupné z: https://tools.ietf.org/html/rfc6797

[5] Prodromou, A brief history of TLS/SSL. *Acunetix* [online]. 2017 [cit. 27. 10. 2017]. Dostupné z: https://www.acunetix.com/blog/articles/history-of-tls-ssl-part-2

[6] Rouse, Secure Sockets Layer (SSL). *TechTarget* [online]. 2017 [cit. 27. 10. 2017]. Dostupné z: http://searchsecurity.techtarget.com/definition/Secure-Sockets-Layer-SSL

[7] Dierks a Allen, The TLS Protocol. *IETF* [online]. 1999 [cit. 27. 10. 2017]. Dostupné z: https://www.ietf.org/rfc/rfc2246.txt

[8] Grigorik, Transport Layer Security (TLS). *High Performance Browser Networking* [online]. 2013 [cit. 30. 10. 2017]. Dostupné z: https://hpbn.co/transport-layer-security-tls

[9] R. Rivest, The MD5 Message-Digest Algorithm. *IETF* [online]. 1992 [cit. 10. 3. 2018]. Dostupné z: https://tools.ietf.org/html/rfc1321

[10] MD5 vulnerable to collision attacks. *Software Engineering Institute* [online]. 2009 [cit. 10. 3. 2018]. Dostupné z: https://www.kb.cert.org/vuls/id/836068

[11] Landman, Ross a Williams, Secure Hashing Algorithms. *Brilliant* [online]. [cit. 10. 3. 2018]. Dostupné z: https://brilliant.org/wiki/secure-hashing-algorithms

[12] P. Nohe, Cipher Suites. *The SSL Store* [online]. [cit. 10. 3. 2018]. 2017 Dostupné z: https://www.thesslstore.com/blog/cipher-suites-algorithms-security-settings

[13] Boneh a Shoup, A Graduate Course in Applied Cryptography. *Tanford University* [online]. 2015 [cit. 10. 3. 2018]. Dostupné z: https://crypto.stanford.edu/~dabo/cryptobook/draft_0_2.pdf

[14] Stephen, SSL and TLS Essentials. *Scribd* [online]. Kanada: 2000 [cit. 12. 3. 2018]. Dostupné z: https://www.scribd.com/document/274335808/SSL-TLS-Essentials

[15] P. Eliza, What is Digital Signature. *EMP Trust HR* [online]. 2017 [cit. 18. 4. 2018]. Dostupné z: http://www.emptrust.com/blog/benefits-of-using-digital-signatures

[16] Trivedi, How Does Email Work?. *How-To Geek* [online]. 2016 [cit. 14. 3. 2018]. Dostupné z: https://www.howtogeek.com/56002/htg-explains-how-does-email-work

[17] J. Klensin, Simple Mail Transfer Protocol. *IETF* [online]. 2008 [cit. 10. 4. 2018]. Dostupné z: https://tools.ietf.org/html/rfc5321

[18] K. Martens, J. Mehnle, Sender Policy Framework. *openspf* [online]. 2008 [cit. 12. 4. 2018]. Dostupné z: http://www.openspf.org/SPF_Record_Syntax

[19] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol. *IETF* [online]. 2008 [cit. 12. 4. 2018]. Dostupné z: https://tools.ietf.org/html/rfc5246

[20] Benchmarks. *cryptopp* [online]. 2017 [cit. 16. 4. 2018]. Dostupné z: https://www.cryptopp.com/benchmarks.html

Zoznam symbolov, veličín a skratiek

AES	Advanced Encryption Standard – algoritmus k šifrovaniu dát
CA	Certification Authority – vydavateľ digitálnych certifikátov
CBC	Cipher Block Chaining – režim pre symetrickú šifru
CSS	Cascading Style Sheets – mechanizmus na vizuálne formátovanie
CLI	Command-line interface – používateľské rozhranie príkazového riadku
CSV	Comma-separated values – súborový formát vo forme čistého textu
DES	Data Encryption Standard – symetrická šifra
DSA	Digital Signature Algorithm – algoritmus digitálneho podpisu
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral – algoritmus výmeny kľúčov
ECDSA	Elliptic Curve Digital Signature Algorithm – podpisová schéma
GCM	Galois/Counter Mode – režim pre symetrickú šifru
HSTS	HTTP Strict Transport Security – bezpečnostný mechanizmus
HTML	Hypertext Markup Language – značkový jazyk určený na vytváranie webových stránok
HTTP	Hypertext Transfer Protocol – komunikačný protokol
HTTPS	Hypertext Transfer Protocol Secure – zabezpečený komunikačný protokol
IETF	Internet Engineering Task Force
MAC	Message Authentication Code – kryptografická funkcia
MD5	Message Digest Algorithm – hashovací algoritmus
MITM	Man in the middle – kryptografický útok
PKI	Public Key Infrastructure – distribúcia verejných kľúčov
RC4	Rivest Cipher 4 – prúdová šifra
RSA	Rivest–Shamir–Adleman – algoritmus šifrovania a podpisovania
SHA	Secure Hash Algorithm – hashovací algoritmus
SMTP	Simple Mail Transfer Protocol – protokol na prenos pošty
SSL	Secure Sockets Layer – kryptografický protokol
TCP	Transmission Control Protoco – protokol transportnej vrstvy
TLS	Transport Layer Security – kryptografický protokol
TSP	Trust Service Provider – poskytovateľ dôveryhodných služieb
UDP	User Datagram Protocol – protokol transportnej vrstvy
URL	Uniform Resource Locator – identifikátor zdroja
VoIP	Voice over IP – protokol na prenos hlasu
3DES	Triple Data Encryption Standard – bloková symetrická šifra

Zoznam príloh

A Návod na inštaláciu Dockeru

V tejto prílohe je popísaný návod na inštaláciu nástroja *Docker*. Na oficiálnej stránke je možné nájsť rôzne typy inštalácií pre určité operačné systémy. Odkaz na oficiálnu stránku je: `https://docs.docker.com/install/`.

V prvom kroku si užívateľ vyberie typ inštalácie pre svoj operačný systém, ktorý používa. Nasleduje samotná inštalácia, pričom užívateľ má možnosť si vybrať z niekoľkých spôsobov inštalácie. Po úspešnej inštalácií je možné vyskúšať funkčnosť nástroja jednoduchým príkazom:

```
$ sudo docker run hello-world
```

Docker automaticky začne sťahovať *image* hello-world zo svojho repozitára a následne ho spustí.

Docker je nastavený tak, že všetky príkazy vyžadujú *root* práva. Ak sa chce užívateľ vyhnúť písaniu sudo pred príkazy *Docker*, musí pridať svoje užívateľské meno do skupiny *Docker*.

```
$ sudo usermod -aG docker ${USER}
```

Týmto skončila inštalácia nástroja *Docker*. Ďalej je potrebné nainštalovať *Docker Compose*, ktorý pracuje s *Docker engine* a umožňuje multi-kontajnerové spúšťanie aplikácií *Docker*. Návod na inštaláciu pre rôzne operačné systémy sa nachádza na oficiálnej stránke: `https://docs.docker.com/compose/install/`.

B Obsah přiloženého CD

Na CD sa nachádza aplikácia vytvorená v programovacom jazyku Go. Samotný projekt je testovaný v operačnom systéme Linux, ale nie je problém ho spustiť aj na inom ľubovoľnom systéme. Okrem toho sú v projekte priložené testovacie sady domén. Súčasťou CD je aj elektronická forma práce.

- `/` ..... koreňový adresár priloženého CD
  - `1. bachelor` ..... hlavný priečinok s projektom
    - `access` ..... funkcie pre prácu s databázou
      - `certificate.go`
      - `server.go`
    - `cmd` ..... spustenie aplikácie
      - `main.go`
    - `core` ..... pomocné funkcie
      - `mock.go`
      - `template.go`
    - `environment` ..... definícia stálych premenných
      - `constants.go`
      - `variables.go`
    - `models` ..... implementácia modelov
      - `analysis.go`
      - `charts.go`
      - `server.go`
      - `worker-pool.go`
    - `service` ..... funkcie pre celý proces analýzy
      - `content.go`
      - `email-pool.go`
      - `parser.go`
      - `service.go`
      - `stats.go`
      - `web-pool.go`
    - `static` ..... súbory pre grafické zobrazenie výsledkov
    - `files` ..... priložené sady domén
      - `cz-sk-domains.csv`
      - `email-providers.csv`
      - `web-servers.csv`
    - `.gitignore`
    - `docker-compose.yml` ..... implementácia docker buildu
    - `Dockerfile`
    - `Gopkg.lock`
    - `Gopkg.toml`
    - `index.html` ..... html šablona pre zobrazenie výsledkov
    - `Makefile` ..... spúšťanie aplikácie
  - `2. Elektronická verzia` ..... elektronická verzia bakalárskej práce
    - `bakalárska-práca.pdf`